Curriculum Learning:
from Human Strategies to Learning Dynamics

Tianyi Zhou

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2022

Reading Committee:

Jeffrey Bilmes, Chair

Ali Farhadi

Kevin Jamieson

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Curriculum Learning:
from Human Strategies to Learning Dynamics

Tianyi Zhou

Chair of the Supervisory Committee:
Professor Jeffrey Bilmes
Electrical and Computer Engineering

The recent success of machine learning (ML), or "the third wave of artificial intelligence (AI)", is built upon computational methods from the fields of optimization and statistics, the availability of large-scale training data and computational power, and partial imitation of human cognitive functions such as convolutional networks. However, current ML techniques can be critically inefficient and prone to imperfect data in practical applications, e.g., when the data are noisy, unlabeled, imbalanced, or contain redundancy, biases, covariate shift, etc. On the other hand, human learning is more strategic and adaptive in planning and selecting training content for different learning stages. Comparing to ML techniques that repeat training on random mini-batches of the same data over all stages, human learning exhibits great advantages in efficiency and robustness when addressing those practical challenges. Therefore, how to develop a strategic “curriculum” for ML becomes an important challenge for bridging the gap between human intelligence and machine intelligence.

Curriculum learning has been first introduced as a data selection method applied to different learning stages based on human-learning strategies, e.g., selecting easier samples at first and gradually adding more and harder ones later. However, the properties of training materials that

humans utilize to design a curriculum are not limited to hardness but can also cover diversity, consistency, representativeness, incentives, impact or utility to future training, etc. In ML, it is challenging to develop efficient and accurate score functions measuring these properties and their contributions to the final/later learning goal. Moreover, given the score functions, it is still an open challenge for a curriculum strategy to plan multiple training stages and adjust the selection criterion adaptive to each stage.

Another primary challenge in curriculum learning is the deficiency of principle and theoretically motivated formulations for the joint optimization of model parameters and the curriculum. Without such formulations, it is difficult to relate selection criteria and score functions to the potential objectives of curriculum learning, e.g., training progress, generalization performance, etc. So it is hard to explain when and why a curriculum can improve ML. Moreover, when developing curriculum learning algorithms, the planning and scheduling of selection criteria for different learning stages need to be designed specifically for different ML applications, e.g., semi-supervised learning, ensemble learning, etc. In order to achieve a practically effective algorithm, it is also important to study whether and how to incorporate existing techniques developed for the specific application with the curriculum.

This thesis aims at addressing the key challenges above. It consists of four parts. In Part I, we introduce several novel formulations for curriculum learning. For example, we can translate human learning strategies to discrete-continuous optimizations and jointly optimize the model and the curriculum over the course of training, as shown in Chapter 2 and Chapter 5. We can also derive an analytic form of the weights or scores from a novel objective for curriculum learning, as shown in Chapter 3 and Chapter 4. Moreover, we discuss several potential formulations in Chapter 6 for future research. In Part II, we take a deep dive into the score function design that plays a significant role in curriculum learning. For example, the diversity of selected data plays a vital role in reducing

redundancy and encouraging early-stage exploration. Besides diversity, we mainly focus on a new class of score functions in Chapter 8, which is based on the training dynamics of a sample over the whole history instead of its instantaneous feedback at a specific step. Comparing to the widely-applied instantaneous scores, they significantly reduce the extra computation required by score evaluations and they are more accurate in allocating the most informative training samples due to their distinguishable dynamic patterns.

In Part III, we build practical curriculum learning algorithms based on the developed formulations and score functions. These algorithms cover several important machine learning problems including supervised learning, semi-supervised learning, noisy-label learning, ensemble learning, etc. In the algorithm for each problem, we study and compare different planning or scheduling strategies that determine how the selection criterion change across learning stages. We justify the effectiveness of the proposed scheduling strategies by detailed empirical analyses and comparisons. In addition, to achieve state-of-the-art performance on each problem, we investigate the interactions between the curriculum and the existing techniques for each problem and then combine their strengths in the algorithmic designs. In Part IV, on each application problem's benchmark datasets, we evaluate our methods and conduct an extensive experimental comparison with a variety of strong baselines. Our methods equipped with the designed curricula consistently bring improvement in both the training efficiency and the final test accuracy in all applications. It is worth noting that the curricula show more significant advantages on more challenging applications with imperfect data such as semi-supervised learning and noisy-label learning.

In Chapter 18, we summarize the main contributions of this thesis. In addition to the proposed formulations, score functions, and algorithms for curriculum learning, we also highlight our efforts on bridging the gaps and combining the strengths of human heuristics, theoretical formulations, and empirical algorithms in a line of work. In addition, we list several potential research directions

to explore in the future work, which can significantly expand the current schemes and application fields of curriculum learning and improve our in-depth understanding of the training dynamics in machine learning as well as its connections to human education and cognition.

TABLE OF CONTENTS

Page

List of Figures v

List of Tables xiii

Chapter 1: Introduction 1
1.1 Curriculum Learning in Human Cognition and Education 1
1.2 Curriculum Learning in Machine Learning 3
1.3 Bridging Machine Curricula with Human Curricula 7
1.4 Main Elements of Curriculum Learning 9
1.5 Main Categories of Curriculum Learning 12
1.6 Methodologies for Curriculum Learning 14
1.7 Thesis Outline 15
1.8 Previously Published Work 21

Part I: Problem Formulations for Curriculum Learning 22

Chapter 2: Continuous-Discrete Hybrid Optimization 23
2.1 Min-min Optimization 23
2.2 Minimax Optimization 25
2.3 Discussion 28

Chapter 3: Reweighing Data through Tilted Loss 29
3.1 Noisy-Label Learning 30
3.2 Optimization Formulation of Robust Curriculum Learning 32
3.3 Data Selection Curriculum of Robust Curriculum Learning 33

Chapter 4: Optimizing Training Dynamics 36

4.1 Optimizing Training Dynamics of Regression . . . 37
4.2 Optimizing Training Dynamics of Classification . . . 40

Chapter 5: Model-Data Matching Curriculum for Diverse Ensemble Learning . . . 42
5.1 Learning a Diverse Ensemble of Expert Models . . . 42
5.2 Motivations of Diverse Ensemble Evolution . . . 45
5.3 Related Work . . . 46
5.4 Diverse Ensemble Evolution: Formulation . . . 49

Chapter 6: Other Formulations . . . 55
6.1 Online Learning and Multi-Armed Bandit . . . 55
6.2 Reinforcement Learning . . . 60
6.3 Meta-Learning or "Learning to Learn" . . . 62

Part II: Scoring Functions for Curriculum Learning . . . 64

Chapter 7: Instantaneous Feedback . . . 66
7.1 Hardness . . . 66
7.2 Subset Diversity . . . 67

Chapter 8: Scores based on Training Dynamics . . . 69
8.1 Dynamic Instance Hardness (DIH) . . . 71
8.2 Scores derived from Optimizing Training Dynamics . . . 80
8.3 Time Consistency of Unlabeled Data . . . 90
8.4 Scores computed from Training Dynamics of Clean and Noisy Labels . . . 99

Part III: Algorithms for Curriculum Learning . . . 106

Chapter 9: Minimax Curriculum Learning (MCL) . . . 107
9.1 Planning and Scheduling of Learning Stages in MCL . . . 108
9.2 Minimax Curriculum Learning Algorithm . . . 109
9.3 Submodular Maximization as a Subroutine . . . 111
9.4 Conditions at Convergence . . . 118
9.5 Practical and Heuristic Improvements . . . 120
9.6 Appendix . . . 123

Chapter 10: Dynamic Instance guided Hardness Curriculum Learning (DIHCL) 128
10.1 A "Free" Curriculum based on Dynamic Instance Hardness 129
10.2 Practical DIHCL with Weighted Sampling 131

Chapter 11: Dynamics-optimized Curriculum Learning (DoCL) 133
11.1 Three Data Selection Criteria using DoCL scores 134
11.2 Dynamics-optimized Curriculum Learning (DoCL) Algorithm 141

Chapter 12: Time-Consistency Curriculum for Semi-Supervised Learning 145
12.1 Training Objective for Semi-Supervised Learning 145
12.2 Time-Consistent Semi-Supervised Learning (TC-SSL) Algorithm 149
12.3 Practical and Heuristic Improvements 151

Chapter 13: Robust Curriculum Learning (RoCL) 153
13.1 Robust Curriculum Learning (RoCL) Algorithm 154
13.2 Compound Scheduling in RoCL Algorithm 155

Chapter 14: Diverse Ensemble Evolution 157
14.1 Ensemble Optimization with Data-Model Marriage 157
14.2 Theoretical Properties 159
14.3 Curriculum towards a Diverse Ensemble with Complementary Expertise 161
14.4 Appendix 164

Part IV: Applications of Curriculum Learning 168

Chapter 15: Supervised Learning 169
15.1 Minimax Curriculum Learning (MCL) 169
15.2 Dynamic Instance Hardness guided Curriculum Learning (DIHCL) 178
15.3 Dynamics-optimized Curriculum Learning (DoCL) 182
15.4 Appendix 189

Chapter 16: Weakly-Supervised Learning 201
16.1 Time-Consistency Curriculum for Semi-Supervised Learning 201
16.2 Noisy-Label Learning by Robust Curriculum Learning (RoCL) 213

Chapter 17: Diverse Ensemble Learning 227
17.1 Experimental Setting 227
17.2 Aggregation Methods using an Ensemble of Models 228
17.3 Experimental Results 230

Chapter 18: Conclusions and Future Directions 237
18.1 Summary of Contributions 237
18.2 Open Challenges and Future Directions 244

Bibliography 250

LIST OF FIGURES

Figure Number Page

1.1 The main elements and main categories of curriculum learning. 9

5.1 Data-Model Marriage as Bipartite Matching. 50

8.1 **Top:** DIH (running mean of loss) vs. **Bottom:** instantaneous loss of 50 randomly selected samples from CIFAR10 on WideResNet-28-10. 73

8.2 **LEFT:** Averaged prediction-flip and **RIGHT:** losses (mean and std.) of the three groups of samples partitioned by a DIH metric (i.e.,running mean of prediction-flip) computed at epoch 10 (top), 40 (middle), and 210 (bottom) during training a WideResNet-28-10 on CIFAR10. Early DIH (epoch 40) can already predict the forgettable and memorable samples at much later stages (epoch 210). The failed partition based on epoch 10 shows the importance of sufficient exploration needed for DIH to accurately measure hardness over time. 74

8.3 **LEFT:** Entry $A_{i,j}$ $(i < j)$ is the percentage of shared samples between the 10k samples with the **largest** DIH computed in epoch $15i$ and epoch $15j$. **RIGHT:** Entry $A_{i,j}$ $(i < j)$ is the percentage of shared samples between the 10k samples with the **smallest** DIH computed in epoch $15i$ and epoch $15j$. It shows that both the hard and easy samples in the future are predictable using the DIH values computed in early epochs. 75

8.4 The three strategies for DIH on 10 hard and 10 easy samples, each that have been randomly sampled from the top 10k samples with the largest/smallest DIH at Epoch 40. 77

8.5 Correlation between DIH in two training experiments using different random seeds. 78

8.6 **LEFT:** Averaged prediction-flip and **RIGHT:** losses (mean and std.) of the three groups of samples partitioned by a DIH metric (i.e.,running mean of prediction-flip) computed at epoch 10,40 and 60 during training WideResNet-28-10 on CIFAR10 with **random labels**. In this setting, the random (but wrong) labels will be remembered very well after some training, and DIH in early stages loses the capability to predict the future DIH, i.e., they can only reflect the history but not the future. This characteristic of DIH might be helpful to detect noisy data. 81

8.7 **LEFT:** Averaged prediction-flip and **RIGHT:** losses (mean and std.) of the three groups of samples partitioned by a DIH metric (i.e.,running mean of prediction-flip) computed at epoch 10, 40, 140 and 210 during training **a smaller CNN** on CIFAR10. It shows that the difference of memorable and forgettable samples is not sufficiently obvious until very late training epochs, e.g., after epoch-140. 82

8.8 **TOP:** Averaged prediction-flip and **RIGHT:** losses (mean and std.) of the three groups of samples partitioned by a DIH metric (i.e., running mean of prediction flip) computed at epoch 40 when using **exponential decaying learning rate** (instead of cyclic cosine annealing rate) across epochs (cycles). DIH exhibits similar properties on identifying hard and easy samples for neural nets to learn. . . . 83

8.9 Losses (mean and std.) of the three groups of samples partitioned by a DIH metric (i.e., running mean of prediction flip) computed at epoch 40 when 40% of labels are randomly changed to another wrong class (i.e., **40% symmetric noises on labels**. We also show the losses on the clean samples with correct labels and noisy samples with wrong labels, where the former exhibit lower DIH than the latter. Hence, DIH is robust to label noises and can identify the hard and easy samples, which are mainly composed of the clean and noisy data respectively in this scenario. 83

8.10 **Top:** DIH (running mean of loss) vs. **Bottom:** instantaneous loss of 10 samples randomly selected from the top 10k samples with the largest(red) and the smallest(blue) DIH at epoch 40 of training of WideResNet-28-10 on CIFAR10 (the same as Figure 8.4. It shows that for each individual sample from the two groups, DIH smoothly decreases while the corresponding instantaneous loss is much noisier. 84

8.11 Computed time-consistency and confidence at epoch 100. The x-axis shows the validation samples selected using different thresholds on the two metrics (normalized to $[0, 100]$). The y-axis reports correct v.s. incorrect predictions over the selected samples. 95

8.12 Computed time-consistency (top) and confidence (bottom) at epoch 100. Select the top 1000 and bottom 1000 validation samples based on the two metrics. Compare the moving average of true class probability of the selected samples across epochs. 96

8.14 Compute time-consistency and confidence at epoch 50. Select samples in the validation set based on different thresholds of the two metrics and report the number of correct v.s. incorrect predictions in the selected samples. The two metrics are normalized to $[0, 1]$ range. 96

8.13 Scatter plot and correlation (cyan lines) of the true class's probability (y-axis) for incorrectly predicted validation samples with TC (**left**) or confidence (**right**) on the x-axis at epoch 100. . . . 97

8.15 Compute time-consistency (a) and confidence (b) at epoch 50. Select the top 1000 and bottom 1000 samples in the validation set based on the two metrics. Compare the moving average of true class probability of the selected samples across training epochs. 97

8.16 Compute time-consistency and confidence at epoch 60. Plot the incorrectly predicted samples in the validation set as a scatter plot with the selection metric as the x-axis and the true class probability as the y-axis. 98

8.17 Compute time-consistency and confidence at epoch 200. Select samples in the validation set based on different thresholds of the two metrics and report the number of correct v.s. incorrect predictions in the selected samples. The two metrics are normalized to $[0, 1]$ range. 98

8.18 Compute time-consistency (a) and confidence (b) at epoch 200. Select the top 1000 and bottom 1000 samples in the validation set based on the two metrics. Compare the moving average of true class probability of the selected samples across training epochs. 99

8.19 Compute time-consistency and confidence at epoch 200. Plot the incorrectly predicted samples in the validation set as a scatter plot with the selection metric as the x-axis and the true class probability as the y-axis. 100

8.20 Dynamic patterns (mean±std) of instantaneous metrics (top) and exponential moving average (EMA) metrics (bottom) when applying Algorithm 1 that alternates between supervised learning on given labels and self-supervision on pseudo labels. *Larger gap between curves in each plot is better.* Symmetric noise is defined in the beginning of Section 16.2.2. We use cross entropy for supervised loss $\ell(\cdot, \cdot)$ in Eq. (8.24) and 0-1 loss for $\ell(\cdot, \cdot)$ in consistency loss Eq. (8.25) with $m = 7$. 103

11.1 Test set accuracy of WideResNet-28-10 (instantaneous model) and its exponential moving average (EMA model) during the course of training when using the three data selection curricula. 135

11.2 *Baseline1 alternates between the highest and lowest scored samples*: Dynamics (mean±std) for (**Top**) the score $\hat{a}(i)$ (Eq. (8.5)) of the selected samples, (**Middle**) DoCL objective (Eq. (4.12)) values of unselected samples, and (**Bottom**) output true-class probabilities for unselected samples. We split the unselected samples in each epoch into two groups with the largest/smallest DoCL objective values. 136

11.3 *Training with highest-scored (Baseline2) vs. lowest-scored (Baseline3) samples*: Dynamics (mean±std) for (**Top**) the score $\hat{a}(i)$ (Eq. (8.5)) of the selected samples, (**Middle**) DoCL objective (Eq. (4.12)) values and (**Bottom**) output true-class probabilities for unselected samples with the largest DoCL objective values. 137

11.4 Test set accuracy of WideResNet-28-10 (instantaneous model) and its exponential moving average (EMA model) during the course of training when using the three data selection curricula. 139

11.5 *Training alternates between the highest and lowest scored samples*: Dynamics (mean±std) for (**Top**) the score $\hat{a}(i)$ (Eq. (8.5)) of the selected samples, (**Middle**) the DoCL objective (Eq. (4.12)) values of unselected samples, and (**Bottom**) the output true-class probability for unselected samples. We split the unselected samples in each epoch into two groups with the largest/smallest DoCL objective values. 139

11.6 *Training with highest-scored vs. lowest-scored samples*: Dynamics (mean±std) for (**Top**) the score $\hat{a}(i)$ (Eq. (8.5)) of the selected samples, (**Middle**) the DoCL objective (Eq. (4.12)) values and (**Bottom**) the output true-class probability for unselected samples with the largest DoCL objective values. 140

13.1 Illustration of Eq. (13.1) and visualization of our choice for $g(\cdot)$ and the resulted τ_t when $T = 50$. We use $g(\cdot) = \tanh(\cdot)$ (which can be other "S"-shape functions) and $\sigma = 0.95$ in our experiments. Here, we map the points on the black curve in the left plot to the points on the red curve in the right plot. Each gray point on the bottom of the left plot is from the T evenly spaced x-coordinates between the x-interval $[g^{-1}(-\sigma), g^{-1}(\sigma)]$. We scale them to the T t-coordinates in the bottom of the right plot (i.e., $t = 1, 2, \cdots, 50$), which associates with T τ_t values represented by the red points between $[\tau_1, \tau_T]$ on the red curve. 156

15.1 Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD). 172

15.2 Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on CIFAR10 (grey curves represents 10 random trials of SGD). 173

15.3 Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on Fashion-MNIST (grey curves represents 10 random trials of SGD). 174

15.4 Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on STL10 (grey curves represents 10 random trials of SGD). 174

15.5 Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on SVHN (grey curves represents 10 random trials of SGD). 175

15.6 Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on MNIST (grey curves represents 10 random trials of SGD). 175

15.7 Number of distinct labeled samples ever needing loss gradient calculation vs. number of training batches for News20 (left), CIFAR10 (middle) and MNIST(right) (grey curves represents 10 random trials of SGD). 176

15.8 Number of distinct labeled samples ever needing loss gradient calculation vs. number of training batches for Fashion-MNIST (left), STL10 (middle) and SVHN (right) (grey curves represents 10 random trials of SGD). 176

15.9 Training DNNs by using DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch SGD on Food-101, Aircraft, CIFAR-10 and FMNIST. We use "Diverse" to denote DIHCL that further reduces S_t by applying submodular maximization for Eq. (10.2). We report how test accuracy changes during training. 179

15.10 Wall clock training time of DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch on ImageNet (top) and Food-101 (bottom). 180

15.11 Comparison of DIHCL variants on training WideResNet-28-10 on CIFAR10. 181
15.12 Training DNNs by using DoCL, DoCL-NR (DIHCL), SPL [116], MCL [248], and random mini-batch SGD on 4 datasets, i.e., CIFAR10, CIFAR100, SVHN and Fashion MNIST. We report how the test accuracy changes with the number of training batches for each method. 184
15.13 Training DNNs by using DoCL, DoCL-NR (DIHCL), SPL [116], MCL [248], and random mini-batch SGD on 5 datasets, i.e., ImageNet, Food101, FGVC Aircraft, Stanford Cars and Birdsnap. We report how the test accuracy changes with the number of training batches for each method. 185
15.14 Ablation study and sensitivity analysis of hyperparameters for DoCL when applied to train WideResNet-28-10 on CIFAR10 (top) and CIFAR100 (bottom). 187
15.15 Regression (knowledge distillation with L2 loss on pre-softmax logits) by DoCL, DoCL-NR, SPL [116], MCL [248], and random mini-batch SGD for transferring the knowledge of a pre-trained ResNeXt-29 8x64d (34.4M parameters) to ResNet-18 (11.2M parameters) on CIFAR10 (top) and CIFAR100 (bottom). 188
15.16 Training error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD). 190
15.17 Training loss vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD). 190
15.18 Test loss vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD). 191
15.19 Training loss vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on CIFAR10 (grey curves represents 10 random trials of SGD). 191
15.20 Test loss vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on CIFAR10 (grey curves represents 10 random trials of SGD). 192
15.21 Training error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on MNIST (grey curves represents 10 random trials of SGD). 192
15.22 Training error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on Fashion-MNIST (grey curves represents 10 random trials of SGD). 193
15.23 Training error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on STL10 (grey curves represents 10 random trials of SGD). 193

15.24 Training error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on SVHN (grey curves represents 10 random trials of SGD). 194

15.25 Training DNNs by using DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch SGD on 3 datasets, i.e., CIFAR10, CIFAR100 and STL-10. We use "Diverse" to denote DIHCL that further reduces S_t by applying submodular maximization for Eq. (10.2). We report how the test accuracy changes with the number of training batches for each method, and the (**log-scale**) wall-clock time for 1) the entire training and 2) the submodular maximization part in DIHCL with diversity and MCL. 196

15.26 Training DNNs by using DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch SGD on 3 datasets, i.e., SVHN, Fashion MNIST and Kuzushiji MNIST. We use "Diverse" to denote DIHCL that further reduces S_t by applying submodular maximization for Eq. (10.2). We report how the test accuracy changes with the number of training batches for each method, and the (**log-scale**) wall-clock time for 1) the entire training and 2) the submodular maximization part in DIHCL with diversity and MCL. 197

15.27 Training DNNs by using DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch SGD on 3 datasets, i.e., ImageNet, Food-101 and Birdsnap. We report how the test accuracy changes with the number of training batches for each method, and the wall-clock time for 1) the entire training and 2) the submodular maximization part in MCL. 198

15.28 Training DNNs by using DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch SGD on 2 datasets, i.e., FGVC Aircraft and Stanford Cars. We report how the test accuracy changes with the number of training batches for each method, and the wall-clock time for 1) the entire training and 2) the submodular maximization part in MCL. 199

16.1 Test accuracy (%) during the training of small WideResNet and large WideResNet by MixMatch and TC-SSL on the four splittings of CIFAR10. 210

16.2 Test accuracy (%) during the training of the large WideResNet by MixMatch and TC-SSL on the two splittings of CIFAR100. 211

16.3 Precision and recall (%) of pseudo labels produced by the model during training for the unlabeled samples selected by TC-SSL. We also provide the percentage (%) of the selected samples in the ground set of unlabeled data $\mathcal{U}$. 212

16.4 **Ablation study**: RoCL vs. its variants during the training of ResNet34 on **CIFAR10** containing **60% symmetric noises** on labels. 221

16.5 **Ablation study**: RoCL vs. its variants during the training of ResNet34 on **CIFAR10** containing **80% symmetric noises** on labels. 221

16.6 **Ablation study**: RoCL vs. its variants during the training of ResNet34 on **CIFAR100** containing **60% symmetric noises** on labels. 222

16.7 **Ablation study**: RoCL vs. its variants during the training of ResNet34 on **CIFAR100** containing **80% symmetric noises** on labels. 222

16.8 RoCL (Algorithm 8) vs. RoCL$_{Base}$ without any curriculum (Algorithm 1) during the training of ResNet34 on **CIFAR10** containing **60% symmetric noises** on labels. 225

16.9 RoCL (Algorithm 8) vs. RoCL$_{Base}$ without any curriculum (Algorithm 1) during the training of ResNet34 on **CIFAR10** containing **80% symmetric noises** on labels. 225

16.10 RoCL (Algorithm 8) vs. RoCL$_{Base}$ without any curriculum (Algorithm 1) during the training of ResNet34 on **CIFAR100** containing **60% symmetric noises** on labels. 226

16.11 RoCL (Algorithm 8) vs. RoCL$_{Base}$ without any curriculum (Algorithm 1) during the training of ResNet34 on **CIFAR100** containing **80% symmetric noises** on labels. 226

17.1 Compare DivE^2 with Bagging(upper row) and RandINIT(lower row) in terms of test accuracy (%) vs. number of training batches on CIFAR10, CIFAR100, Fashion-MNIST and STL10, with $m = 10$ and $k = 3$. 228

17.2 Test accuracy (%) per class on each single model from the ensemble trained by Bagging(left, after 18750 total training batches), RandINIT(middle, after 18750 total training batches) and DivE^2 (right, after 18249 total training batches) on Fashion-MNIST. This figure reflects the expertise of each model on different classes. Comparing to Bagging and RandINIT, the models learned by DivE^2 show diverse and complementary expertise. 232

17.3 Compare DivE^2 with Bagging(left column) and RandINIT(right column) in terms of test accuracy (%) vs. number of training batches on CIFAR10, with $m = 10$ MobileNetV2 models trained, and using different k values ($k = 3, 5, 7$ from top to bottom) for aggregation. 233

17.4 Compare DivE^2 with Bagging(left column) and RandINIT(right column) in terms of test accuracy (%) vs. number of training batches on CIFAR100, with $m = 10$ ResNet18 models trained, and using different k values ($k = 3, 5, 7$ from top to bottom) for aggregation. 234

17.5 Compare DivE^2 with Bagging(left column) and RandINIT(right column) in terms of test accuracy (%) vs. number of training batches on Fashion-MNIST, with $m = 10$ modified LeNet5 models trained, and using different k values ($k = 3, 5, 7$ from top to bottom) for aggregation. 235

17.6 Compare DivE^2 with Bagging(left column) and RandINIT(right column) in terms of test accuracy (%) vs. number of training batches on STL10, with $m = 10$ CNN models trained, and using different k values ($k = 3, 5, 7$ from top to bottom) for aggregation. 236

18.1 Summary of the six curriculum learning methods proposed in this thesis: (1) For each method, we briefly summarize its four components, i.e., its formulation, score, selection criterion, and scheduling strategy; (2) We highlight the main novelty of each method using the color for the name of the method; (3) We illustrate the connections between different methods and the ideas/techniques shared between them; (4) The background color of each method represent its targeted machine learning problem. 238

LIST OF TABLES

Table Number Page

15.1 Test error (%) for different methods (SGD shows the lowest error out of 10 random trials). . . . 169

15.2 Total time (secs.) of MCL($\Delta > 0, \lambda > 0, \gamma > 0$) and time only on WS-SUBMODULARMAX. . . . 171

15.3 The test accuracy (%) achieved by different methods training DNNs on 11 datasets (without pre-training). We use "Loss, dLoss, Flip" to denote the 3 choices of DIH metrics based on (A), (B), and (C) respectively. In all DIHCL variants, we apply diversity (and greedy submodular maximization using the lazier-than-lazy-greedy procedure [153]) for Eq. (10.2) on only the datasets CIFAR10, CIFAR100, STL10, SVHN, KMNIST, and FMNIST. In this case, the first T_0 warm-start epochs of DNN training was used to also produce the feature extractor $z(x)$ to instantiate the facility location function. The other datasets did not employ diversity, and we leave that to future work. For each dataset, the best accuracy is in blue, the second best is red, and third best green. . . . 180

15.4 The test accuracy (%) achieved by random mini-batch SGD (Random), SPL, MCL, DoCL-NR (DIHCL) and DoCL in training DNNs on 9 datasets (without pre-training). In MCL, DoCL-NR and DoCL, we apply lazier-than-lazy-greedy [153] for Eq. (11.1) on CIFAR10, CIFAR100, SVHN and FMNIST. DoCL achieves the highest test accuracy over all 9 datasets. 182

15.5 Details regarding the datasets in MCL experiments. . . . 189

15.6 Parameters of MCL (Algorithm 2) and its variants for different datasets. . . . 189

15.7 Details regarding the datasets and training settings (#Feature denotes the number of features after cropping if applied), "lr_start" and "lr_target" denote the starting and target learning rate for the first episode of cosine annealing schedule, they are gradually decayed over the remaining epochs. . . . 194

15.8 Details regarding the datasets and training settings (cont.) . . . 195

15.9 Details regarding the datasets and training settings (#Feature denotes the number of features after cropping if applied), "lr_start" and "lr_target" denote the starting and target learning rate for the first episode of cosine annealing schedule, they are gradually decayed over the rest episodes. . . . 200

16.1 Test error rate (mean±variance) of SSL methods training a small WideResNet and a large WideResNet on **CIFAR10**. Baselines: Pseudo Label [122], Π-model [181], VAT [155], Mean Teacher [209], MixMatch [22], ReMixMatch [21]. . . . 205
16.2 Test error rate (mean±variance) of SSL methods training WideResNet-28-135 on **CIFAR100**. Baselines: SWA [8], MixMatch [22]. . . . 207
16.3 **Ablation Study:** test error rate (mean±variance) of TC-SSL variants for training WideResNet-28-135 on CIFAR10: "no consistency" removes the consistency loss in Eq. (12.2); "no contrastive" removes the contrastive loss in Eq. (12.3); "no PseudoLabel" removes the cross entropy loss for unlabeled data in Eq. (12.5); "no TC-selection" replaces Line 8-9 of Algorithm 7 with uniform sampling. . . . 207
16.4 Test error rate of several methods training CNNs of different #params on **STL10**. Baselines: CutOut [52], IIC [97], MixMatch [22]. . . . 209
16.5 Accuracy (%) evaluated on WebVision and ILSVRC2012 validation sets for DNNs trained by noisy-label learning methods on mini-WebVision training set (first 50 classes), which contains **real-world web-label noises**. . . . 216
16.6 Test accuracy (%) of **RoCL** applied with different loss functions on CIFAR10 corrupted by $\{60\%, 80\%\}$ **symmetric(uniform)** noises (CE-cross entropy). . . . 218
16.7 Test accuracy (%) of noisy-label learning methods on CIFAR10/100 corrupted by **symmetric(uniform)** label noises of different levels. All the baselines' results are from the original papers or the following-up works. There are two formats of these reported results: "mean±variance" of 5 trials and single-trial accuracy. . . . 219
16.8 Test accuracy (%) of noisy-label learning methods on CIFAR10/100 corrupted by **asymmetric(class-dependent)** noises of 3 levels. All the baselines' results are from the original papers or the following-up works. . . . 220
16.9 **Ablation study:** Test accuracy (%) of RoCL variants with one part removed/changed when applied to CIFAR10/100 corrupted by **symmetric(uniform)** label noise. . . . 220
17.1 Total time (secs.) of DivE^2 and time only on SUBMODULARMAX. . . . 230
17.2 The highest test accuracy (%) achieved by different combinations of ensemble training and aggregation methods on four datasets, with $k = 3$. DivE^2 usually requires less training time than others to achieve the highest accuracy. The best non-cheating test accuracy (i.e., not Top-k Oracle) is highlighted below. . . . 231
17.3 Details regarding the datasets. . . . 232

ACKNOWLEDGMENTS

I am deeply grateful to my advisor, Professor Jeffery Bilmes, whose insightful vision, mathematical intuition, practical ideas, thoughtful discussions, high standards, and tireless pursuit of excellence in everything are truly inspiring. Jeff guided me into the field of submodularity with its applications for machine learning and taught me about not only their fundamental knowledge and motivations but also the methodologies for research and problem solving. I feel extremely lucky to work with Jeff on a number of exciting projects and problems. His priceless advice about problem formulation, rigorous thinking, professional writing, and presentation are critical to my future career as a machine learning researcher. I particularly enjoy the discussions with Jeff and our group meetings led by Jeff, which always bring me new insights and more profound understandings on even well-known problems. In research projects, I learned and benefited a lot from Jeff about how to capture the key ideas and motivations of a work, how to think from both the big pictures and the details, and how to balance the theoretical and practical motivations. In paper writing, Jeff helped me find a variety of mistakes and he is very patient in guiding me to fix them. Besides research, I enjoy the discussions with Jeff about cooking, cultures, travels, etc., and I also enjoy his jokes. Overall, Jeff is an amazing Ph.D. advisor and a supportive friend in life. Thank you, Jeff!

I would like to thank my thesis committee, Professor Ali Farhadi, Professor Kevin Jamieson, and Professor Zaid Harchaoui, for their excellent suggestions, insightful discussions, and constructive questions. I learned a great deal from their work as well. Their wonderful comments greatly help improving this thesis.

I would also like to thank Professor Ben Taskar, Professor Carlos Guestrin, and Professor Emily

Fox, who provided important help and support to me especially when I started my Ph.D. at UW as a junior Ph.D. student.

I would not be at UW today studying machine learning today if not for Professor Dacheng Tao, who introduced me to this field 14 years ago, believed in my potential, and invited me to work with him for years. Dacheng's guidance and support during my early research career is very important and has an immense positive impact on my later researches. I am very grateful to Dacheng for the opportunity and his support for so many years.

I would like to express my appreciation and gratitude to all my collaborators - Shengjie Wang, Wenruo Bai, Chandrashekhar Lavania, Lilly Kumari, Wei Kai, Rishabh Iyer, John Halloran, Sunil Thulasidasan, Arnav Das, and Gantavya Bhatt from UW MELODI Lab; Yuchen Jin, Dianqi Li, Xin Yang, Yuanyuan Yang, Jieyu Zhang, Liangyu Zhao, and Prof. Arvind Krishnamurthy from UW CSE; Tian Li and Prof. Virginia Smith from CMU; Prof. Guodong Long, Prof. Jing Jiang, and Prof. Chengqi Zhang from University of Technology Sydney; Prof. Meng Fang from University of Liverpool; Yanchao Sun, Jiahao Su, and Prof. Furong Huang from University of Maryland; Prof. Shirui Pan from Griffith University; Prof. Tongliang Liu from University of Sydney; Prof. Xinmei Tian from USTC; Yi Mao and Weizhu Chen from Microsoft; Matthew Brown, Boqing Gong, Ming-Hsuan Yang, Yin Cui, Jiahui Yu, Lu Jiang, Utsav Prabhu, Arsha Nagrani, Xuanyi Dong, Le Hou, Xinying Song, and Denny Zhou from Google, for their brilliant ideas, thoughtful discussions, hard works, and dedication. Without their efforts and support, it is impossible to accomplish many amazing projects. I would also like to thank my mentors during my internship at Microsoft Research and Yahoo! Labs: Lin Xiao, Hua Ouyang, Yi Chang, who helped me to broad my interests in different areas and keep supporting and encouraging me after many years of my internship. I was lucky to work with or help mentor several brilliant students - Tao Shen, Lu Liu, Shuang Ao, Haiyan Zhao, Kaiwen Yang, Shuangtong Li, Yijun Yang, Yue Tan, Wensi Tang, Jie Ma,

Songhua Wu, Yuanzhe Pang, Xiang Li, Chen Liang, Yi-Wen Chen, Megha Nawhal, Yibin Lei, and Yu Cao. Their fascinating ideas, in-depth thoughts, and hard works always impress me.

Many thanks to my friends, who make my life at Seattle more interesting and fun. Although the list of their names does not fit in these lines, I am grateful to all of you for your accompany, support, and sharing of the moments of your lives with me.

Finally, I would like to dedicate the thesis to my family for their unbending support, endless love, and tremendous encouragement for years. It was not an easy journey, but all the joy and love you brought to me gave me great courage and strength in pursuing my Ph.D. study and research career. I am very grateful for your accompany, support, and sacrifices. Thank you with all my heart!

DEDICATION

To my family, mentors, and friends for their unending support along the journey.

Chapter 1

INTRODUCTION

1.1 Curriculum Learning in Human Cognition and Education

A curriculum is a planned sequence of learning experiences, materials, strategies, activities, tasks, etc., which are deliberately designed by educators or teachers based on their experience in educating students. It plays an significant role in human learning and is intended to improve the efficiency of students and maximize their learning outcomes. Curricula are ubiquitous in our daily lives from preschool pedagogy, classroom instructions, workplace training, online tutorials, to career planning. A student trained with (1) curricula developed by more experienced teachers and/or (2) curricula better aligned with the student's learning progress can usually learn fast and excel in their future careers. Moreover, curricula bridge generations by transferring the meta-knowledge of "how to learn" or "learning to learn" from one generation to another. In this way, the broad application of curricula has profoundly shaped the course of human history, civilization, and the development of science & technology. The new generations inherit not only the knowledge of the ancestors but also benefit from their developed curricula for improving their learning skills, efficiency, and cognitive functions, e.g., remembering, understanding, analyzing, reasoning, planning, multi-tasking, and creativity.

Conventional curricula are usually lecture-based and designed mainly according to teachers' experiences, which may lack adaptive alignments with students' learning progress and targeted outcome in the future. Recently, problem-based [85, 53] and outcome-based curricula [196] have been studied as more effective strategies to integrate basic concepts into practical problem solving, develop self-directed learning skills, foster increased retention of knowledge, and strengthen

students' intrinsic motivation. In problem-based curricula, the engagement of students in small discussion groups and contextual learning in practical projects are emphasized, leading to an inquisitive style of learning, longer memory of learned knowledge, and "intentional" life-long learners. Outcome-based curricula aim at maximizing the students' capability and achievement after being taught by the curricula. They are usually developed in a backward-design manner [234], i.e., starting from the outcome objectives and then working backwards to address the content, topics, strategies, and materials in the curricula. For different students or learning stages, the learning objectives for the outcome can vary according to the levels in Bloom's Taxonomy. These new methodologies for curricula design show great potentials on improving the learning efficiency as well as strengthening the motivation of students and meeting their needs.

In principal, a curriculum design can be described by its inputs, outputs, and its objective for the outcome. The inputs may include teachers' experiences, prior knowledge, existing curricula, etc., as well as the students' feedback, grades, progress reports, and other possible forms of evaluations at different learning phases. The outputs cover the planning of organization of courses, contents of lessons, assigned tasks and problems, adjustments of learning strategies, etc., for future learning stages. The objective can be the efficiency and performance of the students on achieving certain goals at different levels of Bloom's Taxonomy, fairness among different students, the capability of applying learned skills to unseen problems, or a combination of multiple objectives. Thereby, a general curriculum can adjust its outputs based on its inputs in order to maximize the outcome defined by the objective(s). It can be either determined in an online manner during interactions with students or in an offline manner based on collected experiences, or by alternating between the online and offline settings. Hence, the choices of inputs, outputs and the objective play essential roles in determining the curriculum, and the learning efficiency and outcomes vary dramatically for different curricula. How to find an optimal curriculum design is still an open challenge in human education. Moreover, the above discussion about inputs, outputs, and objectives also implies some

connections between curriculum design and machine learning/teaching and sheds insights into developing curricula for machine learning tasks.

1.2 Curriculum Learning in Machine Learning

Machine learning (ML) is reshaping the world and people's lives in remarkable ways. It can even surpass humans on certain complicated yet specific tasks. However, its utilization and collection of data during training, when viewed from the perspective of human intelligence, is extraordinarily suboptimal and inefficient. Most ML methods treat samples/tasks equally with repetitive training on all of them for many epochs. On the contrary, human learning is more strategic in selecting or generating the training contents/tasks for different learning stages. Though with a potential risk of introducing biases*, under the guidance of such curricula, humans can quickly grasp knowledge from limited examples even when the data contain noisy observations. In fact, we would never teach a child/student the same way we train a neural network since human learning greatly benefits from education with curricula and learning tasks carefully designed by teachers based on students' feedback and learning progress. This raises several open problems, e.g., can we develop a "currciulum" or an optimal training sequence for ML? Whether and how can we transfer learning strategies from human to ML? Do they effectively improve ML?

For instance, an experienced teacher often starts with representative examples that are easy for students to digest the high-level ideas and then guide them to identify and strengthen their weaknesses by repeatedly practicing on hard ones. If the target is a challenging task, an inspiring mentor usually breaks it down into several feasible yet non-trivial sub-tasks for the trainees to develop new and diverse skills by learning to accomplish these sub-tasks one by one instead of struggling with the original task forever. Moreover, a smart and efficient learner can track the forgetting curve of learned knowledge and review it when memory starts to fade out. Furthermore, humans

*This is an important motivation for human-machine hybrid intelligence, e.g., a curriculum taking into account both the human expert experience and the model training dynamics.

exhibit powerful self-learning capability given noisy and biased feedback from the environment with very limited/weak external supervision. We as humans achieve this by evaluating new tasks/data using learned knowledge and seeking the consistent but informative ones to fasten and expand the knowledge. The evaluation usually results in an efficient exploration strategy in the large space of data/tasks, which usually take into account the combination of curiosity, novelty, diversity, and experience from previous learning progress. Moreover, humans learn fast in collaborations with others via knowledge sharing/transferring between different learners, who can learn faster on some topics but slower on some others. In a teamwork environment, each team member typically gets assigned tasks in her/his expertise region. Meanwhile, every task needs to be handled to benefit the whole group, forcing members to develop diverse yet complementary expertise. These strategies constitute an essential part of human intelligence that is however not known how to be coded in ML and optimization algorithms on machines.

Curricula for ML can have diverse forms and affect on different phases or components of ML. Though not often claimed as a curriculum, learning rate schedule is probably the simplest and most common form of curriculum in ML, which controls the learning pace at different stages. The design of learning rate scheduling is essential to convergence of stochastic gradient descent and online learning algorithms, which are the cornerstones of many ML applications. In the practical side of deep learning, tuning the choice of learning rate schedule can result in significant improvement on the efficiency and generalization performance, e.g., cyclical learning rate [192] that resets to a large learning rate periodically in order to escape from and visit more local minimums, and the application of warm starting epochs of increasing learning rate [217] at the very beginning of training. Another example of widely used curricula in ML and statistics is the "homotopy (or path-following) method" [166, 57], which starts from solving a simpler approximation of a targeted hard problem and gradually deforms this problem towards the original problem. During the homotopy, a path of solutions to a sequence of problems with increasing difficulty is achieved consecutively,

where each problem can be solved efficiently by warm starting from the solution for the previous simpler problem. In optimization, this strategy is sometimes referred as continuation method [1], which exhibits advantages in reducing the impact of local minima within neural networks [16].

More sophisticated forms of curriculum learning strategies can be described as mechanisms that determine a sequence of data/tasks/assignments used for training. Early curriculum learning (CL) [106, 12, 197] work shows that feeding an optimized sequence of training sets (i.e., a curriculum), that can be designed by a human expert [19], into the training algorithms can improve the models' performance. CL was later extended to strategies that automatically select samples solely according to the feedback from training progress. Self-paced learning (SPL) [116, 207, 201, 208] chooses the curriculum based on hardness (e.g., per-sample loss) during training. SPL selects samples with smaller loss, and gradually increases the subset size over time to cover all the training data. Self-paced curriculum learning [101] combines the human expert in CL and loss-adaptation in SPL. SPL with diversity (SPLD) [100] adds a negative group sparse regularization term to SPL and increases its weight to increase selection diversity. Other selection criteria have also been studied [71, 74]. However, most of these criteria are developed from heuristics and might not necessarily be directly related to the original training objective. Some of them suffer from hyperparameter sensitivity, e.g., a threshold on loss values. Although the ultimate goal of CL is to find an optimal sequence of training contents, CL criteria are usually built upon relatively simple heuristics instead of being derived from a rigorous mathematical formulation. Some recent work [102, 59] resorts to an additional model to directly generate selection results but they require training another model using non-stationary feedback from the ongoing training process, e.g., via reinforcement learning, which might be more challenging and costly to solve than the original problem.

Although curriculum learning can be designed to control different components of the learning process, dynamic data selection is still the most widely discussed form of curriculum learning algorithms. Data selection has been studied in other ML settings as well. Active learning (AL) [188,

230, 11] allows there to be an interaction between machines and (often human) annotators, where the former can iteratively select samples and query their labels from the latter. It aims to reduce the labeled sample complexity (or annotation cost), and therefore it usually prefers the most uncertain/noisy samples [48, 186, 49, 50] — this also, however, can make the learning susceptible to adversarially chosen noise on a small number of samples. Diversity modeling was introduced to AL in [230]. It uses submodular maximization to select diverse training batches from the most uncertain samples. However, changing the diversity during the learning process as a curriculum has not been investigated in AL as far as we know.

Boosting [185, 65], as an ensemble method, aims to compose a strong learner from sequentially trained weak learners, each trained on a weighted dataset that emphasizes the samples found difficult by the predecessors. Specifically, boosting assigns weights to all samples, with larger weights given to samples having larger loss measured by an aggregation of previously trained models. Both active learning and boosting favor samples that are difficult to predict, since they are purportedly the most informative to learn. Curriculum learning (CL) can benefit from the criteria developed in active learning and boosting but its setting is different: (1) CL does not presume the availability of human annotators so it can only use the data in their original forms (labeled or unlabeled) as collected; (2) CL is not restricted to training an ensemble of weak learners; (3) CL is featured by varying criteria for different learning stages.

Machine teaching (MT) [106, 259, 168, 137] focuses on extracting the smallest "teaching" subset of data that can be used to train a model to produce similar performance as when all the data is used. A recent line of work [138, 139] studies iterative machine teaching (IMT) that allows iterative interactions between the teacher and student via sequential selection of subsets. MT and IMT are different from CL: (1) they assume that the teacher knows the optimal model — CL does not make this assumption; (2) their objective is to minimize the distance between a student model to the optimal one, which is different from CL.

The selection criteria of these methods were developed for various learning settings and hypothesis class assumptions, and thus can sometimes be contradictory. For example, active learning and boosting both favor difficult-to-learn samples, while many CL methods prefer easy-to-learn samples [101, 116]. Although selection criteria are often partially adaptive to per-sample feedback during training, they are not designed to directly accelerate the learning process. Moreover, similar to human learning, curriculum learning has the capability to adjust the selection criteria for different learning stages. But how to change the criteria accordingly is still an open problem. In addition, similar to experienced teachers who take the students' learning history into account, it is natural to relate curriculum learning with observed training dynamics, which is not fully explored in existing work.

1.3 Bridging Machine Curricula with Human Curricula

In analogy to curriculum design in human learning, we can also consider different choices of inputs, outputs, and objective for curriculum development in machine learning. Such an analogy can substantially extend current curriculum learning in ML. In particular, the teaching and learning strategies in human education can be transferred to the ML curriculum as input prior knowledge. For example, a commonly adopted strategy in human learning, which is also effective in training ML models, is to start from learning easier data/tasks before addressing more challenging ones [116]. Another human learning strategy, which encourages the curiosity and/or diversity in exploration, is also found critical in various machine learning problems, e.g., online learning, active learning, supervised learning, self-supervised learning, reinforcement learning, etc. How to transfer effective human learning strategies as curricula facilitating ML is still an open problem, and resolving it may bridge some gap between human intelligence and machine intelligence. On the other hand, as in human learning, the engagement of students (i.e., the trained models in ML) can also provide significant information to improve the curriculum. Specifically, the performance of students (ML

models) on different data/tasks over history, i.e., the learning dynamics, can provide important metrics for adjusting the curriculum to match the students' progress over the course of training. Thereby, the curriculum can be adaptive and focus on improving the weakness of the students and their lacked skills.

In machine learning, it is also worthwhile to study different objectives for the curriculum design, e.g., learning efficiency, training loss, robustness to noises or adversarial attacks, generalization to unseen data or domains, transferability to other tasks and environments, etc. The overarching goal of curriculum learning, similar to human teaching, may target different levels of learning objectives from remembering, understanding, applying, analyzing, evaluating, to creating. Although most of the current ML problems focus on the bottom levels, e.g., how to remember facts/concepts, understand them, and apply them in new scenarios, high-level learning objectives might be achievable in the future through newly developed ML technologies and the associated curriculum designs.

1.4 Main Elements of Curriculum Learning

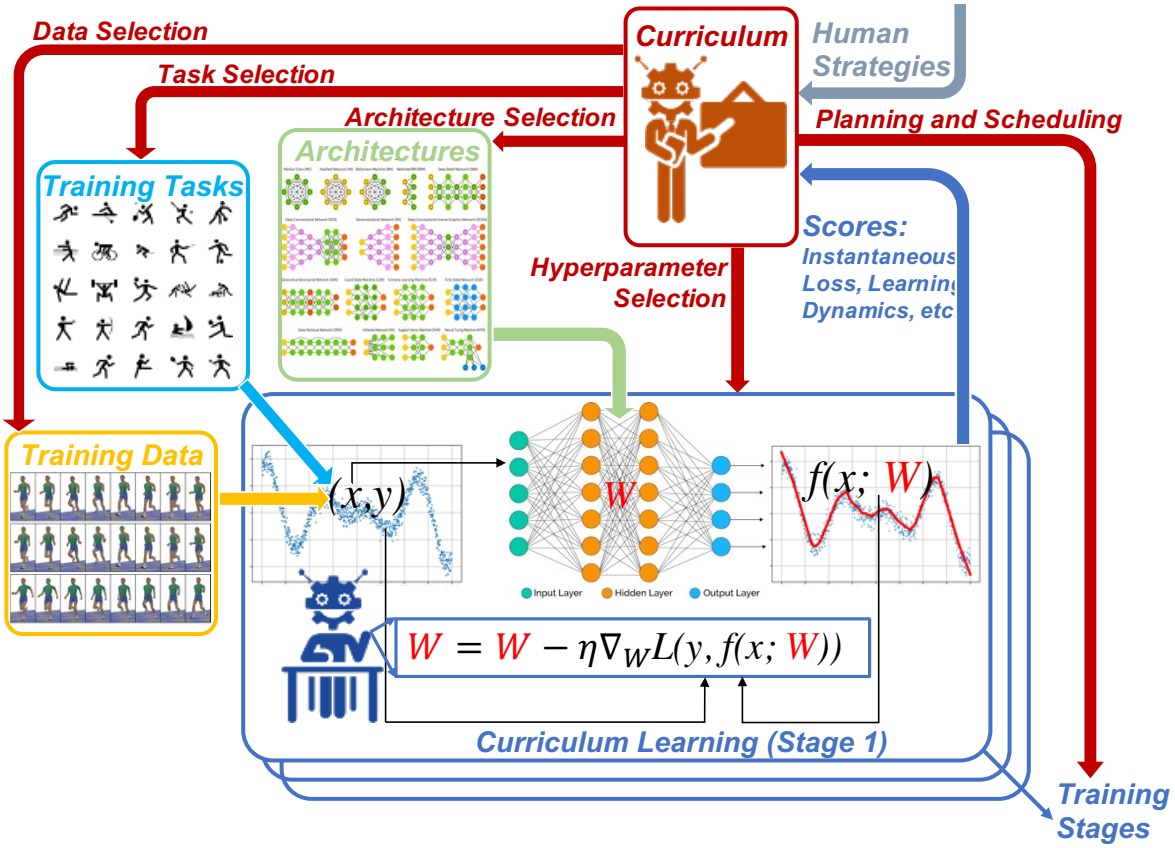

Figure 1.1: The main elements and main categories of curriculum learning.

In conventional machine learning paradigms, the training data (or their distribution), tasks (e.g., input and output spaces), training objectives, hyperparameters, and model architectures are usually determined at the very beginning and then fixed during the rest course of training. In human learning, however, these components can be scheduled and adjusted by teachers or learners in an adaptive manner. In analogy to human learning, curriculum learning has the capability of selecting these components for different training stages and dynamically adjusting them to improve the training process towards achieving better efficiency and generalization performance. In this section, we

discuss the main elements of a curriculum used for the selection of the components defining each learning stage. The design of these elements has a significant impact on the effectiveness of the curriculum.

In a nutshell, as illustrated in Figure 1.1, a curriculum makes a **learning plan** that partitions the whole training process into several stages, each applying a **selection criterion** to choose the learning components according to certain **score functions**, which evaluate the utility of candidate choices for those components. For instance, a plausible planning strategy for a curriculum is to gradually increase the hardness of tasks or training data by increasing the threshold of the task/data selection criterion, which selects the tasks or samples whose hardness scores are below the threshold. In addition, the hardness can be measured by different score functions, e.g., loss, gradient norm, variance of predictions, etc.

Planning. The planning in curriculum learning splits the whole course of training into several consecutive stages that aim at achieving a sequence of progressive goals. Since the plan provides a long-term scheduling of future activities and it needs to be made before training begin, it usually relies on prior knowledge or previous learning experiences. For example, the plan can be "learning from easy to hard" according to human experiences that one can start to learn easier tasks faster in earlier stages and such warm-up stages can improve the future learning of more challenging tasks. More complicated planning strategies may take multiple attributes into accounts and change them jointly, for example, besides solely increasing the hardness, a curriculum can meanwhile increase the total number of samples per stage, decrease the learning rate, and/or improve the representativeness of the selected training data. Moreover, each attribute is usually changed according to a certain schedule function determining how quickly the attribute is increased or reduced over stages, e.g., linear, exponential, or cosine functions[†] are the common choices for the schedule functions.

[†]For example, cosine annealing with a single monotone-decreasing cycle or multiple cycles of oscillation, where used besides in learning rate schedule.

Selection Criterion. In each learning stage of a curriculum, a selection criterion is applied to choose the learning components such as training data or tasks from a ground set of candidates. The simplest and most widely used criterion might be thresholding, e.g., it selects all the data whose scores are above or below a certain threshold (which value for every stage is scheduled by the planning strategy). More generally, a selection criterion can be derived as a solution of an optimization problem that aims at finding the a subset of data, tasks, or training objectives that has the maximal (or minimal) score among all others. When the score for a subset is a modular function that sums up the scores of all elements belonging to the subset, the selection criterion reduces to ranking and top-k selection. However, when the score is not modular and intends to capture relationships between elements, e.g., diversity or redundancy, the optimization problem can be more challenging. In addition, a selection criterion in some scenarios needs to perform exploration-exploitation trade-off in order to achieve more accurate estimate of the scores for underexplored samples.

Score Functions. As discussed above, the selection criterion is usually developed based on a score function that evaluates the utility, hardness, or information of every candidate for the learning components controlled by the curriculum. The design of score functions has been an popular topic in curriculum learning and can lead to critical differences on the final performance. Ideally, a score function is expected to reflect the contribution of each candidate (e.g., a sample or a task) to the future training and thus filters out the redundant or less helpful candidates. For example, a variety of work measures the hardness of each sample by its loss value computed on the model during training, which reflects the weakness of the model that may need to be overcame in the future learning. However, most existing score functions are crafted based on heuristics or empirical observations so it is usually difficult to directly relate them to the final training objective such as the validation loss.

1.5 Main Categories of Curriculum Learning

In general, a curriculum produces a sequence of output actions that control different components of ML over a course of training stages. Therefore, we can categorize different curricula designs according to their output actions. Although not all of them have been fully explored by either current literature of curriculum learning or the new methods proposed in this thesis, they provide important future research directions that can potentially expand the potential of curriculum learning and further bridge the gap between machine learning and human learning. We provide an illustration covering most of those categories in Figure 1.1.

- **Hyperparameter curriculum.** This type of curricula changes the training hyperparameters, e.g., learning rate, regularization weight, batch size, resolution of input images, etc. It plays important roles in practice but usually needs to be hand-tuned by human experts.

- **Data-level curriculum.** This type of curricula selects a subset of data samples for training in each learning stage. There exist different criteria to select data, e.g., hardness, diversity, progress, etc., which can vary across different training stages. For structured data such as audio/video/text sequences, data-level curriculum can be fine-grained and extended to phoneme/frame/token selection. Another example is graph structured data, on which a curriculum can select nodes or edges that are most informative for training a probabilistic graphical model or a graph neural network. Data selection might be the most commonly studied form of curricula in curriculum learning.

- **Task-level curriculum.** This type of curricula selects or creates tasks to train the model in different learning stages. These tasks can have either different outputs or goals. For example, a complicated task can be decomposed into a sequence of easier sub-tasks or sub-goals that are more efficient to learn, learning to categorize data into coarse classes at first can help

the consequent training of a classifier over fine classes, a model trained on lower-resolution images can be used to warm start the training on higher-resolution images, etc.

- **Domain(Environment)-level curriculum.** This type of curricula progressively changes the data domain (e.g., from sketch to photo of objects) or the interactive environment (e.g., in reinforcement learning) over the course of training. In both cases, the input data distribution changes. For example, an image generative model might first learns to synthesize simple sketches before targeting photo realistic images, a robot might first learn to walk on a flat terrain before navigating on a steep slope or staircase with obstacles, etc. It can be developed to improve the generalization and robustness of ML models or AI agents when adapted to different domains or unseen environments.

- **Model-level curriculum.** This type of curricula imitates the morphology evolution of species in nature over billions of years. More complicated and versatile morphology is developed to adapt to the changes or new challenges emerging in the environment. In ML, it is analogous to gradually changing the architecture of an ML model or the morphology of an embodied AI agent in order to develop diverse key skills addressing different challenges in a sequence of targeted tasks, data, or environments. For efficient training, we may start from training a small model in a simplified environment (domain) and gradually grow the model to excel in more intricate environments. Such a curriculum can also be developed for more efficient neural architecture search of a compact model.

- **Assignment Curriculum.** This type of curricula selects and assigns training data (or teacher models) to different student models (or modules/layers in a single model). This problem can be usually formulated as partitioning based on models' (students') feedback. On the other hand, the partitioning result also determines the training contents for the next stage. The assignment is adaptive to the learning progress of all models and can vary over the course

of training. For example, it may encourage the diversity of student models/modules and accelerate their training by matching the most informative data/teachers and can be applied in ensemble learning, federated learning, decentralized learning, multi-agent learning, etc.

1.6 Methodologies for Curriculum Learning

One primary goal of this thesis is studying how to transfer sophisticated and adaptive human-learning strategies to generate curricula for ML tasks in order to improve their training efficiency, generalization performance, and robustness, especially when the training data are imperfect, e.g., containing noises, redundancy, or a large portion of unlabeled samples. The settings are not only limited to supervised learning. In fact, we will show that curriculum learning can bring much more significant improvements to some ML problems than the supervised learning mainly considered in previous works. Another goal of this thesis is studying the relationship between curriculum learning and training dynamics. On the one hand, we will design empirical studies to verify whether the training dynamics contain important information for curriculum design. On the other hand, we will attempt to establish a principle formulation for curriculum learning that aims to optimize the training dynamics of data drawn from the targeted distribution.

Although data selection and related problems have been widely studied in active learning, machine teaching, curriculum learning, experimental design, etc., their targeted ML settings and models are limited, and the algorithms are not competitive enough in training deep neural networks (DNNs) on large-scale datasets. This thesis mainly focuses on three novel methodologies below that have been less explored but are essential for creating practical curricula improving the current ML schemes.

- **Continuous-discrete optimization** [248, 252, 253]. A curriculum is a sequence of discrete decisions on selected samples/tasks/environments optimized on the fly with continuous model parameters over the course of training. Hence, we need to develop new mathematical objec-

tives for the online selections and novel formulations for continuous-discrete optimization. They raise open challenges in developing efficient optimization algorithms.

- **Training dynamics of ML models** [253, 255, 256, 251, 225, 224]. A good teacher can adjust and optimize the curriculum based on students' feedback, progress, and learning history. Analogously, the training dynamics of ML models also contain critical information to the auto-generation of an efficient curriculum, and optimizing the training dynamics may lead to a principled mathematical formulation for curriculum learning. Therefore, we empirically and theoretically study the training dynamics of DNNs under different curricula, which result in practical selection criteria and innate connections to existing deep learning theories.

- **Diversity and curiosity** [248, 252, 250, 227] are two crucial driven forces of human civilization and scientific exploration. In the curriculum generation for ML, they also play important roles in selecting representative and informative data/tasks, reducing the data bias/imbalance, improving exploration effectiveness, and fostering different yet complementary expertise on multiple models. We study functions and metrics to measure diversity and curiosity in various learning settings, develop optimization tools to promote diversity or curiosity during the training process, and study the interplay between them with the feedback from training dynamics. Submodular optimization [24] plays a vital role in promoting and formulating different types of diversity.

1.7 Thesis Outline

This thesis consists of four parts. In Part I, we will introduce several possible formulations of curriculum learning, which are associated with different objectives that a curriculum aim to optimize. Although many curriculum learning methods are proposed directly as data selection algorithms, algorithms derived for solving rigorously formulated problems are usually better motivated and

can provide more in-depth insights on how the resulted curricula improve the model training. In Part II, we will discuss the development of different score functions for curriculum learning: while a great number of previous works focus on instantaneous feedback, we introduce several scores that can be computed or derived from training dynamics and analyze their advantages in identifying the importance training samples. In Part III, we will introduce several curriculum learning algorithms that are built upon the score functions and problem formulations as well as several practical strategies. In Part IV, we will show some detailed applications of curriculum learning in supervised learning, semi-supervised learning, noisy-label learning, and ensemble learning, in which we demonstrate the advantages of our proposed algorithms when compared with previous curriculum learning approaches and methods that do not utilize any curriculum.

In the following, we summarize all the chapters of this thesis.

Chapter 2: Curriculum learning with data selection can be formulated as a joint optimization of both the model parameters and the selected data subset, i.e., a continuous-discrete hybrid optimization that can be challenging to solve in practice. In this chapter, we discuss and compare two main categories of such formulation, i.e., min-min optimization and minimax optimization, where the former always focuses on the easiest samples with the smallest loss, the latter tends to primarily learn the most challenging samples and optimize an upper bound of the average loss.

Chapter 3: We introduce another type of formulation of curriculum learning built upon a novel class of loss functions, i.e., tilted loss that aggregates per-sample losses by a LogSumExp function with a temperature coefficient. In contrast to the continuous-discrete optimization, this formulation does not explicitly optimize the data subset but relaxes the selection problem as a first-order continuous optimization with a weighted combination of per-sample gradients, where the weight for each sample measures its importance in the curriculum. Moreover, we can combine multiple tilted losses to encode a composition of multiple selection criteria. As a practical example, we present a

formulation for robust curriculum learning with noisy labels: the curriculum aims at selecting both the data with clean given labels and the data with correct pseudo labels.

Chapter 4: We investigate a novel formulation of curriculum learning whose goal is to optimize the training dynamics on the targeted data distribution by training data selection. In particular, we study the continuous-time learning progress towards the ground truths under the model dynamics induced by the gradient flow computed on the selected samples, for two categories of learning tasks, i.e., regression and classification. Maximizing the learning progress results in a selection criterion and a score function that take both the hardness and the representativeness into account.

Chapter 5: We provide a formulation for the curriculum learning of a diverse but complementary ensemble of expert models. The proposed formulation, "diverse ensemble evolution (DiVE2)", aims at effectively learning a diverse ensemble by dynamically adjusting the data-model matching during the course of training. We formulate the problem as a continuous-discrete optimization. The discrete part is a generalized bipartite graph matching including two diversity terms in its objective and under two partition-matroid constraints. It assigns a partition of training data to each model. On the other hand, the continuous part trains each model using the assigned data.

Chapter 6: We briefly discuss other possible formulations for curriculum learning, e.g., online learning, contextual or non-stationary bandits, meta-learning, reinforcement learning. These formulations usually require to conduct sufficient exploration over all possible candidates or train a curriculum on multiple learning experiences/trajectories.

Chapter 7: We provide a brief introduction to the instantaneous feedback widely used as score functions in curriculum learning. We mainly discuss two types of instantaneous feedback, i.e., hardness and subset diversity, which depend on instantaneous evaluations of the model and represen-

tations during the course of training. These scores have been utilized to select the most important training data for each learning stage.

Chapter 8: We propose four score functions that are computed or derived from the training dynamics for individual samples. Unlike the scores based on instantaneous feedback, these score functions do not require to re-evaluate every candidate in each step but instead leverage the patterns of observed training dynamics as the indicators of important samples for future learning stages. Moreover, the score functions and dynamic patterns are specifically developed for different learning tasks, e.g., supervised learning, semi-supervised learning, and noisy-label learning (which combines both the supervised learning and self-supervised learning).

Chapter 9: We introduce a curriculum learning algorithm, "minimax curriculum learning (MCL)", based on a minimax formulation and a score function that performs trade-off between the hardness and diversity of the selected subset in each step. A primary novelty of MCL is its planning strategy for the hardness and diversity of training data over different stages. Specifically, MCL starts from learning a few diverse and representative samples and then gradually turns its focus to hard and confusing samples near the classification boundaries. Hence, its earlier stages result in a quick sketch of a relatively well-performed model while the later stages fine-tune the margins to address the most challenging samples. In addition, when the loss is convex w.r.t. the model and the score function is submodular w.r.t. the subset, MCL provably solves the minimax optimization with an approximation and convergence guarantee.

Chapter 10: We develop a curriculum learning algorithm based on a training dynamics driven score function, i.e., "dynamic instance hardness (DIH)", which maintains the exponential moving average (EMA) of a instantaneous feedback (e.g., the loss or the change of loss/prediction between two steps). The "DIH guided curriculum learning (DIHCL)" algorithm focuses on samples with

large DIH, which indicate a sharp local minima achieved on their loss landscapes that result in easy forgetting during the future training. Equipped with certain exploration strategies, DIHCL only requires the training byproduct to update the DIH of each sample and it improves the learning efficiency by focusing on finding a flat minima for the large DIH samples.

Chapter 11: We develop a curriculum learning algorithm, "dynamics-optimized curriculum learning (DoCL)", based on the score function derived in Chapter 8 to optimize the training dynamics. The score combines the residual and the linear temporal dynamics of the model output on each sample, which measures the hardness and the sharpness of the loss landscape. We motivate DoCL by comparing the model training when using different data partitioned by the score function. The DoCL algorithm encourages the model focusing on the samples at the learning frontier, i.e., those with large loss but fast learning speed, and thus leads to more efficient training progress.

Chapter 12: We propose a curriculum learning algorithm, "time-consistent semi-supervised learning (TCSSL)", for semi-supervised learning mainly built upon two self-supervised learning techniques, i.e., consistency regularization and contrastive loss, which leverage the model outputs as pseudo-labels. TCSSL utilizes a "time-consistency" score to select unlabeled samples whose pseudo labels are consistent over time steps, which improves the quality of pseudo labels used in semi-supervised learning losses. As a result, TCSSL significantly improves the efficiency and final performance of semi-supervised learning by avoiding frequent changes of pseudo labels for the same sample.

Chapter 13: We develop a curriculum learning algorithm, "robust curriculum learning (RoCL)", based on the formulation in Chapter 3 and score functions similar to DIH and time-consistency. RoCL addresses two key challenges in noisy-label learning: (1) detecting the clean/correct labels and (2) correcting the noisy/wrong labels. In particular, RoCL applies a curriculum performing a

smooth transition from supervised learning using correct given-labels (but possibly wrong pseudo labels) to self-supervision using correct pseudo-labels (but possibly wrong given labels). Hence, the curriculum can focus on the most informative samples with the correct labels and considerably improve the efficiency and performance of noisy-label learning.

Chapter 14: We develop a curriculum learning algorithm for "diverse ensemble evolution (DiVE2)", which effectively learns an ensemble of diverse but complementary expert models by dynamically adjusting the data-model matching during the course of training. By gradually changing the constraints and the weights for diversity terms, the data-model matching smoothly alters from "each model selects the easiest samples" to "each sample is assigned to the best-performing model", where the former phase focuses on fostering diverse expertise and the latter phase enforces their expertise to be complementary.

Chapter 15: We apply the three curriculum learning algorithms proposed in Chapter 9-11 to supervised learning on multiple benchmark datasets and compare them with previous curriculum learning methods as well as the methods without using any curriculum. Thorough comparisons demonstrate the advantages of data selection curricula on improving both the training efficiency and the final test accuracy of supervised learning. Moreover, we also investigate different designs of curricula and the importance of main elements in a curriculum via detailed ablation studies.

Chapter 16: We apply TCSSL (Chapter 12) and RoCL (Chapter 13) to two weakly-supervised learning scenarios respectively, i.e., semi-supervised learning and noisy-label learning, in which self-supervised learning plays an important role and its effectiveness significantly relies on the label quality of the selected training data. We evaluate the two algorithms and compare them with existing methods on a spectrum of benchmarks with different ratios for the unlabeled or noisy-label data. The results show that their curricula considerably accelerate the slow convergence of

the two weakly-supervised learning tasks and bring non-trivial improvements to the final model performance.

Chapter 17: On multiple benchmark datasets, we demonstrate the advantages of DivE^2 algorithm (Chapter 14) in training DNN ensembles compared to classical and widely used ensemble methods.

Chapter 18: We summarize the main contributions presented in this thesis and discuss important future research directions based on the current results and insights.

1.8 Previously Published Work

Some of the work presented in this thesis has been published in conference proceedings. The minimax formulation in Chapter 2, the MCL algorithm in Chapter 9, and the MCL experiments in Chapter 15 were published at ICLR 2018 [248]. The DIH score in Chapter 8, the DIHCL algorithm in Chapter 10, and the DIHCL experiments in Chapter 15 were published at NeurIPS 2020 [253]. The formulation of optimizing training dynamics in Chapter 4, the scores derived from optimizing training dynamics in Chapter 8, the DoCL algorithm in Chapter 11, and the DoCL experiments in Chapter 15 were published at AISTATS 2021 [255]. The time-consistency score in Chapter 8, the TCSSL algorithm in Chapter 12, and the semi-supervised learning experiments in Chapter 16 were published at ICML 2020 [251]. The tilted loss based formulation in Chapter 3, the scores for clean and noisy labeled data in Chapter 8, the RoCL algorithm in Chapter 13, and the noisy-label learning experiments in Chapter 14 were published at ICLR 2021 [256]. The DiVE2 formulation in Chapter 5, algorithm 14, and the experiments of ensemble learning in Chapter 17 were published at NeurIPS 2018 [252].

Part I

PROBLEM FORMULATIONS FOR CURRICULUM LEARNING

A widely studied optimization formulation for conventional machine learning without any curriculum is empirical risk minimization (ERM). Given a training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of n samples and loss function $L(y_i, f(x_i, w))$, where $x_i \in \mathbb{R}^m$ represents the feature vector for the i^{th} sample, y_i is its label, and $f(x_i, w)$ is the predicted label provided by a model with weight w, ERM aims at minimizing the loss averaged on the training set, i.e.,

$$\min_{w \in \mathbb{R}^m} \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, f(x_i, w)\right). \tag{1.1}$$

Instead of assign the equal weight to every training sample in the objective, curriculum learning studies the difference among samples on their contributions to different training stages and aims at finding the most informative data that can result in the greatest learning progress. For example, it dynamically adjusts the weights of samples in the objective or select different subsets of samples for different training stages. Therefore, curriculum learning usually solves a different problem as ERM. Specifically, an additional group of variables is usually optimized together with the model parameters, e.g., the subset of selected samples or the weights applied to each sample's loss. Moreover, the objective needs to be redesigned for optimizing the subset or the weights. In this part, we will discuss and propose several problem formulations for curriculum learning.

Chapter 2

CONTINUOUS-DISCRETE HYBRID OPTIMIZATION

A straightforward formulation of curriculum learning is to formulate the data selection as another group of variables optimized together with the model parameters. Since the model parameters are continuous variables and the subset of selected data is a discrete variable, this type of formulation usually needs to solve a continuous-discrete hybrid optimization.

2.1 Min-min Optimization

Inspired by the human interaction between teacher and student, recent studies [106, 12, 197] support that learning algorithms can be improved by updating a model on a designed sequence of training sets, i.e., a curriculum. This problem is addressed in curriculum learning (CL) [19], where the sequence is designed by a human expert or heuristic before training begins. Instead of relying on a teacher to provide the curriculum, self-paced learning (SPL) [116, 207, 201, 208] chooses the curriculum during the training process. It does so by letting the student (i.e., the algorithm) determine which samples to learn from based on their hardness. SPL can be formulated as the following min-min optimization:

$$\min_{w\in\mathbb{R}^m} \min_{\nu\in[0,1]^n} \left[\sum_{i=1}^{n} \nu_i L\left(y_i, f(x_i, w)\right) - \lambda \sum_{i=1}^{n} \nu_i \right]. \tag{2.1}$$

SPL jointly learns the model weights w and sample weights ν, which end up being 0-1 indicators of selected samples, and it does so via alternating minimization. Fixing w, minimization w.r.t. ν selects samples with loss $L(y_i, f(x_i, w)) < \lambda$, where λ is a "hardness parameter" as it corresponds to the

hardness as measure by the current loss (since with large λ, samples with greater loss are allowed in). Self-paced curriculum learning [101] introduces a blending of "teacher mode" in CL and "student mode" in SPL, where the teacher can define a region of ν by attaching a linear constraint $a^T\nu \leq c$ to Eq. (2.1). SPL with diversity (SPLD) [100], adds to Eq. (2.1) a negative group sparse regularization term $-\gamma\|\nu\|_{2,1} \triangleq -\gamma \sum_{j=1}^{b} \|\nu^{(j)}\|_2$, where the samples are divided into b groups beforehand and $\nu^{(j)}$ is the weight vector for the j^{th} group. Samples coming from different groups are thus preferred, to the extent that $\gamma > 0$ is large.

CL, SPL, and SPLD can be seen as a form of continuation scheme [1] that handles a hard task by solving a sequence of tasks moving from easy to hard; the solution to each task is the warm start for the next slightly harder task. That is, each task, in the present case, is determined by the training data subset and other training hyperparameters, and the resulting parameters at the end of a training round are used as the initial parameters for the next training round. Such continuation schemes can reduce the impact of local minima within neural networks [16, 17]. With SPL, after each round of alternating minimization to optimize Eq. (2.1), λ is increased so that the next round selects samples that have a larger loss, a process [106, 208, 12] that can both help avoid local minima and reduce generalization error. In SPLD, γ is also increased between training rounds, increasingly preferring diversity. In each case, each round results in a fully trained model for the currently selected training samples.

The SPL approach starts with a smaller set of easy samples and gradually increases the difficulty of the chosen samples as measured by the sample loss of the model produced by previous round's training. One of the difficulties of this approach is the following: since for any given value of λ the relatively easiest samples are chosen, there is a good chance that the process can repeatedly select a similar training set over multiple rounds and therefore can learn slowly. This is precisely the problem that SPLD address — by concomitantly increasing the desired diversity over rounds, the sample selection procedure chooses from an increasingly diverse set of different groups, as

measured by $\|\nu\|_{2,1}$. Therefore, in SPLD, early stages train on easier not necessarily diverse samples and later stages train on harder more diverse samples.

There are several challenges remaining with SPLD, however. One is that in early stages, it is still possible to repeatedly select a similar training set over multiple rounds since diversity might not increase dramatically between successive rounds. Potentially more problematically, it is not clear that having a large diversity selection weight in late stages is desirable. For example, with a reasonably trained model, it might be best to select primarily the hardest samples in the part of the space near the difficult regions of the decision boundaries. With a high diversity weight, samples in these difficult decision boundary regions might be avoided in favor of other samples perhaps already well learnt and having a large margin only because they are diverse, thereby leading to wasted effort. At such point, it would be beneficial to choose points having small margin from the same region but that might not have the greatest diversity, especially when using only a simple notion of diversity such as the group sparse norm $\|v\|_{2,1}$. Also, it is possible that late stages of learning can select outliers only because they are both hard and diverse. Lastly, the SPL/SPLD min-min optimization involves minimizing a lower bound of the loss, while normally one would, if anything, wish to minimize the loss directly or at least an upper bound. This is a potential pitfall for the min-min optimization since the objective has no impact or constraint on the loss of the hardest samples, which can dominate the empirical risk and thus fail the training.

2.2 Minimax Optimization

We introduce and study minimax curriculum learning (MCL), a new method for adaptively selecting a sequence of training subsets for a succession of stages in machine learning. MCL target at minimizing an upper bound of the empirical risk averaged over all samples, i.e., the loss on a subset of the most challenging samples. To achieve the goal, the subsets are encouraged to be small and diverse early on, and then larger and allowably more homogeneous in later stages. At each stage,

model weights and training sets are chosen by solving a joint continuous-discrete minimax optimization, whose objective is composed of a continuous loss (reflecting training set hardness) and a discrete submodular promoter of diversity for the chosen subset. MCL repeatedly solves a sequence of such optimizations with a schedule of increasing training set size and decreasing pressure on diversity encouragement. We reduce MCL to the minimization of a surrogate function handled by submodular maximization and continuous gradient methods. We show that MCL achieves better performance and, with a clustering trick, uses fewer labeled samples for both shallow and deep models. Our method involves repeatedly solving constrained submodular maximization of an only slowly varying function on the same ground set. Therefore, we develop a heuristic method that utilizes the previous submodular maximization solution as a warm start for the current submodular maximization process to reduce computation while still yielding a guarantee.

We introduce a new form of curriculum learning called minimax curriculum learning (MCL). MCL increases desired hardness and reduces diversity encouragement over rounds of training. This is accomplished by solving a sequence of minimax optimizations, each of which having the form:

$$\min_{w \in \mathbb{R}^m} \max_{A \subseteq V, |A| \leq k} \sum_{i \in A} L\left(y_i, f(x_i, w)\right) + \lambda F(A). \tag{2.2}$$

The objective is composed of the loss on a subset A of samples evaluating their hardness and a normalized monotone non-decreasing submodular function $F : 2^V \rightarrow \mathbb{R}_+$ measuring A's diversity, where V is the ground set of all available samples. A larger loss implies that the subset A has been found harder to learn, while a larger $F(A)$ indicates greater diversity. The weight λ controls the trade-off between hardness and diversity, while k, the size of the resulting A, determines the number of samples to simultaneously learn and hence is a hardness parameter.

It is important to realize that $F(A)$ is not a parameter regularizer (e.g., ℓ_1 or ℓ_2 regularization on the parameters w) but rather an expression of preference for a diversity of training samples. In

practice, one would add to Eq. (2.2) an appropriate parameter regularizer as we do in our experiments (Section 15.1).

Like SPL/SPLD, learning rounds are scheduled, here each round with increasing k and decreasing λ. Unlike SPL/SPLD, we explicitly schedule the number of selected samples via k rather than indirectly via a hardness parameter. This makes sense since we are always choosing the hardest k samples at a given λ diversity preference, so there is no need for an explicit real-valued hardness parameter as in SPL/SPLD. Also, the MCL optimization minimizes an upper bound of the loss on any size k subset of training samples.

The function $F(\cdot)$ may be chosen from the large expressive family of submodular functions, all of which are natural for measuring diversity, and all having the following diminishing returns property: given a finite ground set V, and any $A \subseteq B \subseteq V$ and a $v \notin B$,

$$F(v \cup A) - F(A) \geq F(v \cup B) - F(B). \tag{2.3}$$

This implies v is no less valuable to the smaller set A than to the larger set B. The marginal gain of v conditioned on A is denoted $f(v|A) \triangleq f(v \cup A) - f(A)$ and reflects the importance of v to A. Submodular functions [66] have been widely used for diversity models [136, 134, 13, 171, 69, 92, 25].

Although Eq. (2.2) is a hybrid optimization involving both continuous variables w and discrete variables A, it can be reduced to the minimization of a piecewise function, where each piece is defined by a subset A achieving the maximum in a region around w. Each piece is convex when the loss is convex, so various off-the-shelf algorithms can be applied once A has been computed. However, the number of possible sets A is $\binom{n}{k}$, and enumerating them all to find the maximum is intractable. Thanks to submodularity, fast approximate algorithms [158, 152, 153] exist to find an approximately optimal A. Therefore, the outer optimization over w will need to minimize an approximation of the piecewise function defined by an approximate A computed via

submodular maximization. Our algorithm addressing Eq. (2.2), which will be presented in Chapter 9, follows an optimization strategy alternating between solving the inner and outer optimizations. The experimental results will be presented in Section 15.1.

2.3 Discussion

Minimax formulation has a better guarantee than the min-min formulation since it minimizes an upper bound of ERM instead of a lower bound. However, simply choosing the objective as the max-loss makes the training always focus on the hardest samples, which may contain outliers. Moreover, the hardest samples might introduce an extra bias towards a few confusing classes. In order to overcome these problems, we introduce a diversity term $F(A)$ in Eq. (2.2) that encourages the selected training samples to be representative of the whole training set. This strategy is not limited to a specific formulation or method but can be applied to many other curriculum learning methods that will be introduced later in this thesis. On these methods, adding a diversity objective for data selection consistently brings further improvement across different machine learning tasks.

Chapter 3

REWEIGHING DATA THROUGH TILTED LOSS

In curriculum learning, the loss usually provides critical information and performs as a score function for identifying the most important training samples. Instead of optimizing the subset or sample weights as an additional group of variables, can we construct a novel loss function that can automatically adjust the weight for each sample according to its loss? In this chapter, we study how to apply the "tilted loss" [129] to formulating curriculum learning for noisy-label learning. The "tilted loss" shows several intriguing properties in different learning settings.

Unlike the conventional empirical risk minimization (ERM) with arithmetic average loss, i.e., $1/n \sum_{i=1}^{n} \ell(f(x_i;\theta), y_i)$, tilted empirical risk minimization (TERM) [129] minimizes the tilted loss, which is a LogSumExp loss with an additional temperature parameter, i.e.,

$$\min_{\theta} \frac{1}{\tau} \log \left(\frac{1}{n} \sum_{i=1}^{n} \exp[\tau \ell(f(x_i;\theta), y_i)] \right), \tag{3.1}$$

where θ are the model parameters, $f(x_i;\theta)$ is the model prediction for a sample x_i, $\ell(f(x_i;\theta), y_i)$ is the loss by comparing the prediction with the ground truth label y_i, and τ is the temperature.

The key property of tilted loss facilitating the formulation of curriculum learning is: by changing the temperature τ, tilted loss can approximate both the min-loss (when $\tau \to -\infty$), max-loss (when $\tau \to +\infty$), and any interpolation between them, where the min-loss focuses on the easiest samples with the smallest losses and the max-loss focuses on the hardest ones. Note it reduces to the arithmetic average loss when $\tau \to 0$. Hence, by minimizing the tilted loss with a schedule of varying τ, we can achieve a curriculum controlling the focus of each training stage on samples with

different loss values. Although the rest of this chapter focuses on noisy-label learning, this property can be leveraged to formulate the curriculum learning problems for other application tasks.

3.1 Noisy-Label Learning

The expressive power and high capacity of deep neural networks (DNNs) result in accurate modeling and promising generalization if provided with sufficient data and clean(correct) labels. However, existing studies show that the training process is fragile and can easily overfit on noisy labels [242], which commonly appear in real-world data since precise annotation is not always available or affordable. Hence, it is important to study the training dynamics affected by imperfect labels and develop robust learning strategies that ideally eliminate the negative impact of noisy labels while fully exploiting the information from all the available data.

Numerous approaches have been developed to address this challenge from various perspectives, e.g., loss correction [237, 215, 124, 218, 131], robust loss functions [68, 246, 228, 144] with provable noise tolerance, sample re-weighting [169], model co-teaching [76], and the major topic of this thesis, i.e., curriculum learning [116, 103, 72]. A principal methodology behind a variety of methods is to detect clean labels while removing/downweighing the data with wrong labels, so the model mainly learns from correct labels. A broadly applied criterion is to select the samples with small losses and treat them as clean data. It is inspired by empirical observations that DNNs learn simple patterns first before overfitting on the noisy labels [242, 6]. Several curriculum learning methods utilize this criterion [116, 100], and in each step, select/upweigh samples with small losses. Robust loss functions also suppress the large losses associated with the possibly wrong labels. More recent approaches use mixture models [4] to estimate the distribution of losses for clean and noisy data. However, the widely used cross entropy loss enforces the neural network classifier to predict a class for each sample, even when it is confusing to the classifier. So the classifier can be overconfident on some data with noisy labels so the small loss criterion is not always accurate.

Hence, recent work [211] develops a novel loss function that allows abstention of confusing samples and enables dynamic adaption of abstention rates based on learning progress during training.

However, the instantaneous loss (i.e., the loss evaluated at the current step) of an individual sample is an unstable signal that can rapidly fluctuate due to DNN training's randomness. The error generated by such an unstable metric accumulates when the selected samples are used to train the model producing the losses. Co-teaching methods alleviate this problem by training two DNNs and using the loss computed on one model to guild the other. Also, as the model changes during training, each sample's loss needs to be re-evaluated even when it is not selected, which requires extra inference cost. MentorNet [103] and Data Parameters [184] train an extra model to produce the sample weights or selection results without computing the loss. Furthermore, it may not be efficient to repeatedly train the model only on clean data that consistently have small losses, since the model have already learned, well memorized or overfitted to them.

A primary drawback of training only on clean labels detected is that discarding the whole data pairs (x, y) with wrong labels y removes potentially useful information about the data distribution $p(x)$ [4]. Hence, there has been growing interest in leveraging noisy data. Loss correction methods aim to correct the predicted class probabilities based on an estimated mislabeling probability between classes. Some other methods seek to relabel them by using the model itself (e.g., bootstrapping loss [178]) or another model/mechanism (e.g., directed graphical models, conditional random fields, or CNNs) trained on an additional set of clean data, which, however, is not always available. Self-training and unsupervised learning techniques [176, 22] have also been employed to generate pseudo labels to replace noisy labels [4]. The pseudo labels are optimized together with the model or generated by the model with data augmentations to encourage the output consistency on the same sample's augmentations. Unfortunately, the pseudo labels' quality may vary across different samples and significantly degenerate when the noise ratio is high, or the model fails to produce stable and correct predictions. In such a case, the relabeling error on some samples can be accumulated during

training.

We propose a novel noisy-label learning (NLL) formulation "robust curriculum learning (RoCL)" that performs a smooth transition from supervised learning on data with clean labels to self-supervised learning on data with possibly wrong labels. In NLL, the classification loss (measuring the difference between the model prediction and the given label) and the consistency loss (measuring the difference between the model prediction and the pseudo-label on data augmentations) for each sample can provide effective score functions to identify the samples with correct given labels and correct pseudo-labels, respectively. Specifically, a small classification loss $\ell(f(x_i;\theta), y_i)$ usually indicates the correctness of the given label for sample-i, while a small consistency loss $\zeta(i)$ usually indicates the correctness of the pseudo-label for sample-i. In RoCL, we improve the NLL efficiency by a curriculum that aims to select the most informative data for NLL, which enforces (1) the supervised learning stages mainly focusing on the samples with correct given labels but possibly incorrect pseudo-labels; and (2) the self-supervised learning stages mainly focusing on the samples with correct pseudo-labels but possibly incorrect given labels. Since this chapter mainly focuses on the problem formulations, we will leave the detailed discussion of the two score functions to Section 8.4 and the RoCL algorithm to Chapter 13.

3.2 Optimization Formulation of Robust Curriculum Learning

The optimization formulation of robust curriculum learning (RoCL) aims to minimize a combination of supervised loss and consistency(self-supervised) loss in the following form.

$$\min_{\theta} F(\theta) \triangleq \frac{\lambda}{\tau_1} \log\left(\frac{1}{n}\sum_{i=1}^{n} \exp[\tau_1 \ell(f(x_i;\theta), y_i)]\right) + \frac{1-\lambda}{\tau_2} \log\left(\frac{1}{n}\sum_{i=1}^{n} \exp[\tau_2 \zeta(i)]\right), \quad (3.2)$$

where the consistency loss $\zeta(i)$ is defined in Eq. (8.25) (with t removed). The first term of Eq. (3.2) focuses on the supervised learning loss. The second term of Eq. (3.2) focuses on the consistency loss $\zeta(i)$. We use λ to control the trade-off between the supervised loss and the consistency loss. By

simple algebra, the gradient of $F(\theta)$ at step t is

$$\begin{aligned}\nabla_\theta F(\theta_t) &= \lambda \sum_{i=1}^{n} p'_t(i) \nabla_\theta \ell(f(x_i;\theta_t), y_i) + (1-\lambda) \sum_{i=1}^{n} q'_t(i) \nabla_\theta \zeta_t(i) \\ &= \sum_{i=1}^{n} \mathscr{P}'_t(i) \left[\frac{\lambda p'_t(i)}{\mathscr{P}'_t(i)} \nabla_\theta \ell(f(x_i;\theta_t), y_i) + \frac{(1-\lambda) q'_t(i)}{\mathscr{P}'_t(i)} \nabla_\theta \zeta_t(i) \right] = \mathbb{E}_{i \sim \mathscr{P}'_t(i)} G_t(i), \end{aligned} \tag{3.3}$$

where

$$G_t(i) \triangleq \frac{\lambda p'_t(i)}{\mathscr{P}'_t(i)} \nabla_\theta \ell(f(x_i;\theta_t), y_i) + \frac{(1-\lambda) q'_t(i)}{\mathscr{P}'_t(i)} \nabla_\theta \zeta_t(i), \tag{3.4}$$

and

$$p'_t(i) \triangleq \frac{\exp[\tau_1 \ell(f(x_i;\theta_t), y_i)]}{\sum_{j=1}^{n} \exp[\tau_1 \ell(f(x_j;\theta_t), y_j)]}, \quad q'_t(i) \triangleq \frac{\exp[\tau_2 \zeta_t(i)]}{\sum_{j=1}^{n} \exp[\tau_2 \zeta_t(j)]}, \quad \mathscr{P}'_t(i) = \lambda p'_t(i) + (1-\lambda) q'_t(i). \tag{3.5}$$

For every training step, an unbiased estimator of the gradient in Eq. (3.3) can be achieved by drawing a subset of samples S_t according to $\mathscr{P}'_t(i)$ and averaging their gradients $G_t(i)$ in Eq. (3.4).

3.3 Data Selection Curriculum of Robust Curriculum Learning

In this section, we develop an efficient curriculum for NLL, "Robust Curriculum Learning (RoCL)", based on the formulation in Eq. (3.2). RoCL applies a scheduling of $(\tau_1, \tau_2, \lambda)$, which defines a sequence of problems in the form of Eq. (3.2) with different choices of $(\tau_1, \tau_2, \lambda)$. By solving them sequentially as a curriculum, RoCL performs a smooth transition from supervised learning to self-supervised learning on the most informative data with high-quality labels. Specifically, RoCL utilizes the classification loss as a score to select correct given labels for supervised learning (i.e., clean label detection) and utilizes the consistency loss as a score to select correct pseudo-labels for self-supervised learning (i.e., pseudo-label selection).

In order to further improve the quality of both the given and pseudo labels selected by the curriculum, we replace the loss scores $\ell(f(x_i;\theta), y_i)$ and $\zeta(i)$ with their exponential moving averages

(EMA) $l_t(i)$ and $c_t(i)$ at each step-t, respectively. We provide a detailed study in Section 8.4 showing that the EMA scores are better alternatives to the instantaneous feedback scores when used for data selection in NLL.

Hence, we use $\{p_t(i), q_t(i), \mathscr{P}_t(i)\}$ computed from the EMA scores to replace $\{p'_t(i), q'_t(i), \mathscr{P}'_t(i)\}$ computed from the instantaeous feedback scores. RoCL samples x_i at training step t with probability:

$$\mathscr{P}_t(i) = \lambda p_t(i) + (1-\lambda) q_t(i), \tag{3.6}$$

where $p_t(i)$ and $q_t(i)$ are defined as softmax probabilities, i.e.,

$$p_t(i) \triangleq \frac{\exp[\tau_1 l_t(i)]}{\sum_{j=1}^{n} \exp[\tau_1 l_t(j)]}, \quad q_t(i) \triangleq \frac{\exp[\tau_2 c_t(i)]}{\sum_{j=1}^{n} \exp[\tau_2 c_t(j)]}, \tag{3.7}$$

and $\lambda \in [0, 1]$ controls trade-off between them. Here, $p_t(i)$ is a softmax probability computed from EMA losses for clean data detection: samples with smaller (larger) EMA losses and thus clean (noisy) labels tend to have high probability $p_t(i)$ when τ_1 is negative (positive). Similarly, $q_t(i)$ is computed from EMA consistency loss for pseudo label selection: samples with smaller (larger) EMA consistency loss and thus correct (wrong) pseudo labels tend to have high probability $q_t(i)$ when τ_2 is negative (positive). When $\tau_1, \tau_2 = 0$, the probabilities are uniform, and when $\tau_1, \tau_2 \to +\infty$ $(-\infty)$, the probabilities approximate the max (min) operator.

Either $p_t(i)$ or $q_t(i)$ can be independently employed to select or reweigh samples for noise-label learning. However, selecting samples with high probabilities $p_t(i)$ when $\tau_1 \ll 0$ tends to make the training focus on clean data that the model has already learned, which carries limited new information and prevents exploration. Similarly, selecting samples with high probabilities $q_t(i)$ when $\tau_2 \ll 0$ is not informative since the model outputs are consistently correct for those data, and little progress can be made. We can encourage exploration by manipulating the temperature parameters. By setting τ_1 and τ_2 close to zero, we move towards uniform exploration of all data. A

more effective strategy is to couple the values of τ_1 and τ_2. For example, a negative τ_1 and a positive τ_2 strengthen the preference for clean data that have not been fully exploited and learned by the model. Alternatively, a positive τ_1 with a negative τ_2 emphasizes the noisy data with correct pseudo labels, so relabeling them provides new information in addition to the clean data.

We apply the data selection criterion of Eq. (3.6) in each step and gradually change its parameters $(\tau_1, \tau_2, \lambda)$ over the course of training according to the properties discussed above. We start from a negative τ_1 associated with a positive τ_2 and a large λ ($p_t(i)$ dominates), then gradually increase τ_1 while decrease τ_2 and λ, and end with a positive τ_1, a negative τ_2 and a small λ. In this way, we get a curriculum with a smooth transition between supervised learning of clean data using correct given labels and self-supervision of noisy data using reliable pseudo labels, i.e., minimizing Eq. (8.25). Moreover, the coupling strategy on τ_1 and τ_2 encourages selecting informative samples that the model mostly needs to improve on, i.e., clean data with inconsistent model outputs or noisy data with correct pseudo labels.

Chapter 4

OPTIMIZING TRAINING DYNAMICS

In this chapter, we propose to select training sample subsets that most quickly help the predictions for all samples in the training set get close to their targets (Eq. (4.1)). Unlike previous data selection methods, we directly relate our selection criteria to the changes in the training objective at every step. Specifically, we select samples to maximize the linear dynamics of the model's output along the direction from the current output to its ground truth target at time-t, i.e., the learning speed, in expectation over the data distribution. This provides **a principle formulation of curriculum learning** from which we can derive data selection criteria:

$$\max_{S\subseteq[n],|S|\leq k} \mathbb{E}_{x\sim\mathcal{D}} \left\langle y - f(x), \left.\frac{\partial f(x)}{\partial t}\right|_S \right\rangle. \tag{4.1}$$

where $\mathcal{D}$ is the empirical training data distribution, $[n] = \{1, 2, \ldots, n\}$ is the training index set, $f(x)$ is the model's prediction for x, y is the target for $f(x)$, and the linear dynamics $\frac{\partial f(x)}{\partial t}|_S$ can be thought of as the prediction change for x when training on subset S. In the discrete time, when applying gradient descent of a loss $\ell(y, f(x))$ to update the parameters θ in $f(\cdot)$ with learning rate η, i.e.,

$$\theta' = \theta - \eta \frac{\partial}{\partial \theta} \sum_{i\in S} \ell(y_i, f(x_i)), \tag{4.2}$$

we can approximate the linear dynamics $\frac{\partial f(x)}{\partial t}|_S$ as

$$\left.\frac{\partial f(x)}{\partial t}\right|_S \approx f(x;\theta') - f(x;\theta). \tag{4.3}$$

Following a simple analysis for regression and classification objectives, we reduce the problem to selecting samples with the largest scores in each step, with the score on each sample calculated from its current residual and its linear dynamics under the gradient flow computed on the data distribution $\mathcal{D}$. These two quantities can be estimated directly via byproducts of normal training, and therefore, computing the curriculum of training sets above incurs negligible additional costs.

The ultimate goal of curriculum learning is to find an optimal sequence of training samples that will lead to faster training progress, lower training computation, and better generalization performance. It is, however, prohibitively expensive to directly search for the optimal sequence since the set of candidates grows exponentially with nT, where n is the training set size and T is the number of training epochs. In this section, we reduce curriculum learning to optimizing the per-step training dynamics for samples drawn from the data distribution $\mathcal{D}$. Specifically, at step t, we show how to select a subset $S_t \subseteq [n]$ leading to f making the greatest progress towards the ground truth y in expectation for $x \sim \mathcal{D}$ as per Eq. (4.1). By relating it to a simple analysis of training dynamics, we will show that the problem can be efficiently solved using only pre-existing byproducts of training, as mentioned above. For simplicity, we remove all subscripts denoting the time step, for example, we use S for S_t.

4.1 Optimizing Training Dynamics of Regression

We first consider a regression task that aims to learn a prediction model $f(x;\theta)$ by minimizing the expected ℓ_2 loss $\ell(y, f(x;\theta))$ for x drawn from the data distribution $\mathcal{D}$, i.e.,

$$\min_{\theta} \mathbb{E}_{x\sim\mathcal{D}} \ \ell(y, f(x;\theta)) \triangleq \frac{1}{2}\|y - f(x;\theta)\|_2^2. \tag{4.4}$$

In the following, we will use simplified notations: we will use $f(x)$ and $\ell(x)$ to denote $f(x;\theta)$ and $\ell(y, f(x;\theta))$, respectively. Under the gradient flow (continuous-time gradient descent) computed on a subset $S \subseteq [n]$, we have $\frac{\partial\theta}{\partial t}\big|_S = -\sum_{i\in S} \frac{\partial \ell(x_i)}{\partial \theta} = \sum_{i\in S} -\frac{\partial \ell(x_i)}{\partial f(x_i)} \cdot \frac{\partial f(x_i)}{\partial \theta}$. The corresponding

discrete time change of θ under the gradient descent with learning rate η, according to Eq. (4.2), is

$$\theta' - \theta = -\eta \frac{\partial}{\partial \theta} \sum_{i \in S} \ell(x_i). \tag{4.5}$$

Hence, the gradient flow at time step-t can be approximated by

$$\left.\frac{\partial \theta}{\partial t}\right|_S \approx \frac{\theta' - \theta}{\eta} = -\frac{\partial}{\partial \theta} \sum_{i \in S} \ell(x_i). \tag{4.6}$$

The linear dynamics of the model's output $f(x)$ for any sample x can be represented as

$$\left.\frac{\partial f(x)}{\partial t}\right|_S = \frac{\partial f(x)}{\partial \theta} \cdot \left.\frac{\partial \theta}{\partial t}\right|_S = \frac{\partial f(x)}{\partial \theta} \cdot \sum_{i \in S} -\frac{\partial \ell(x_i)}{\partial f(x_i)} \cdot \frac{\partial f(x_i)}{\partial \theta}. \tag{4.7}$$

For this section, we always assume that the optimization is performed in the continuous time domain, so the gradients, chain-rules and integration are all well-defined, and the derivation holds rigorously. If conducting such analysis in the discrete time, there will be a number of approximations within the reasoning that are needed to be carefully addressed. That being said, the result derived in the continuous time setting cannot be directly applied in practical discrete time gradient descent algorithms. Hence, we will discuss its discretization in Sec. 11.2 when we need to estimate the continuous-time quantities in an algorithmic implementation.

At step t, we aim to find a subset $S \subseteq [n]$ of size $|S| \leq k$ whose gradient flow maximizes the projection of residual $y - f(x)$ on the dynamics $\left.\frac{\partial f(x)}{\partial t}\right|_S$ for all $x \sim \mathcal{D}$, i.e.,

$$\max_{S \subseteq [n], |S| \leq k} \mathbb{E}_{x \sim \mathcal{D}} \left\langle y - f(x), \left.\frac{\partial f(x)}{\partial t}\right|_S \right\rangle. \tag{4.8}$$

Intuitively, the goal is to maximize the momentum of each sample's prediction $f(x)$ moving towards its target y. The dynamics $\partial f(x)/\partial t$ are weighted by their residuals $y - f(x)$ so we achieve faster

decreasing loss for samples with larger residual. Thereby, predictions of different samples ideally can reach their targets at the same time without overshooting. The objective maximizes the dynamics of decreasing the objective in Eq. (4.4), i.e., $\frac{\partial \mathbb{E}_{x\sim\mathcal{D}}\ -\ell(y,f(x;\theta_t))}{\partial t} = \mathbb{E}_{x\sim\mathcal{D}}\left\langle y - f(x;\theta_t),\ \frac{\partial f(x;\theta_t)}{\partial t}\Big|_S\right\rangle$.

One can think that we break down the original problem in Eq. (4.4) into a sequence of sub-problems in the form of Eq. (4.8) over time steps. To verify this, we integrate the objective in Eq. (4.8) over time from $t = 0$ to T, which recovers the objective in Eq. (4.4) (negated, up to a constant):

$$\int_0^T \left\langle y - f(x;\theta_t), \frac{\partial f(x;\theta_t)}{\partial t}\right\rangle dt = \frac{1}{2}\left(\|y - f(x;\theta_T)\|_2^2 - \|y - f(x;\theta_0)\|_2^2\right). \tag{4.9}$$

Since the gradients $\partial\ell(x_i)/\partial\theta$ computed on different samples might have conflicts and cancel out with each other if selected in the same training batch, compared to uniform sampling S, maximizing the dynamics encourages selecting samples with consistent gradients that decrease the expected risk/loss over the data distribution.

To optimize Eq. (4.8), we approximate the expected momentum w.r.t. $x \sim \mathcal{D}$ in Eq. (4.8) by averaging over a finite number of samples D drawn from the data distribution $\mathcal{D}$, i.e.,

$$\begin{aligned}
\mathbb{E}_{x\sim\mathcal{D}}\left\langle y - f(x), \frac{\partial f(x)}{\partial t}\Big|_S\right\rangle \approx & \frac{1}{|D|}\sum_{x\in D}[y - f(x)]^T \frac{\partial f(x)}{\partial \theta}\cdot\sum_{i\in S} -\frac{\partial \ell(x_i)}{\partial f(x_i)}\cdot\frac{\partial f(x_i)}{\partial \theta} \\
= & \frac{1}{|D|}\sum_{i\in S} -\left[\frac{\partial \ell(x_i)}{\partial f(x_i)}\right]^T \cdot \sum_{x\in D}\frac{\partial f(x_i)}{\partial \theta}\left[\frac{\partial f(x)}{\partial \theta}\right]^T [y - f(x)] \\
= & \frac{1}{|D|}\sum_{i\in S}\left[\frac{\partial \ell(x_i)}{\partial f(x_i)}\right]^T \frac{\partial f(x_i)}{\partial \theta}\cdot\sum_{x\in D}\left[\frac{\partial f(x)}{\partial \theta}\right]^T \frac{\partial \ell(x)}{\partial f(x)} \\
= & \frac{1}{|D|}\sum_{i\in S}\left[\frac{\partial \ell(x_i)}{\partial f(x_i)}\right]^T \frac{\partial f(x_i)}{\partial \theta}\cdot -\frac{\partial \theta}{\partial t}\Big|_D \\
= & \frac{1}{|D|}\sum_{i\in S}\left\langle y_i - f(x_i), \frac{\partial f(x_i)}{\partial t}\Big|_D\right\rangle.
\end{aligned} \tag{4.10}$$

4.2 Optimizing Training Dynamics of Classification

We can extend the above analysis of dynamics for regression to the general multi-class classification task, which learns a model $f(x;\theta)$ to minimize the cross entropy loss $\ell_{xe}(x;\theta)$:

$$\min_{\theta}\mathbb{E}_{x\sim\mathcal{D}}\ \ \ell_{xe}(x)\triangleq-\log p(x)[y],\ \ p(x)[y]=\frac{\exp(f(x)[y])}{\sum_{j=1}^{c}\exp f(x)[j]}, \tag{4.11}$$

where c is the number of classes and y is the class label of x. We denote the one-hot encoding of y as $\mathbf{y}$. Similarly, at step t, we aim to find a subset $S\subseteq[n]$ of size $|S|\leq k$ whose resulting gradient flow maximizes the projection of the residual $\mathbf{y}-p(x)$ onto the dynamics $\frac{\partial p(x)}{\partial t}\Big|_S$ for all $x\sim\mathcal{D}$, i.e.,

$$\max_{S\subseteq[n],|S|\leq k}\mathbb{E}_{x\sim\mathcal{D}}\left\langle\mathbf{y}-p(x),\frac{\partial p(x)}{\partial t}\Big|_S\right\rangle. \tag{4.12}$$

Similar to the regression case, we approximate the expectation with samples D drawn from $\mathcal{D}$, i.e.,

$$\begin{aligned}
\mathbb{E}_{x\sim\mathcal{D}}\left\langle\mathbf{y}-p(x),\frac{\partial p(x)}{\partial t}\Big|_S\right\rangle \approx&\frac{1}{|D|}\sum_{x\in D}[\mathbf{y}-p(x)]^T\cdot\frac{\partial p(x)}{\partial f(x)}\cdot\frac{\partial f(x)}{\partial\theta}\cdot\sum_{i\in S}-\frac{\partial\ell_{xe}(x_i)}{\partial f(x_i)}\cdot\frac{\partial f(x_i)}{\partial\theta}\\
=&\frac{1}{|D|}\sum_{i\in S}-\left[\frac{\partial\ell_{xe}(x_i)}{\partial f(x_i)}\right]^T\cdot\sum_{x\in D}\frac{\partial f(x_i)}{\partial\theta}\left[\frac{\partial f(x)}{\partial\theta}\right]^T\left[\frac{\partial p(x)}{\partial f(x)}\right]^T[\mathbf{y}-p(x)]\\
=&\frac{1}{|D|}\sum_{i\in S}\left[\frac{\partial\ell_{xe}(x_i)}{\partial f(x_i)}\right]^T\frac{\partial f(x_i)}{\partial\theta}\cdot\sum_{x\in D}\left[\frac{\partial f(x)}{\partial\theta}\right]^T\left[\frac{\partial p(x)}{\partial f(x)}\right]^T\frac{\partial\ell(x)}{\partial p(x)}\\
=&\frac{1}{|D|}\sum_{i\in S}\left[\frac{\partial\ell_{xe}(x_i)}{\partial f(x_i)}\right]^T\frac{\partial f(x_i)}{\partial\theta}\cdot\sum_{x\in D}\left[\frac{\partial f(x)}{\partial\theta}\right]^T\frac{\partial\ell(x)}{\partial f(x)}\\
=&\frac{1}{|D|}\sum_{i\in S}\left[\frac{\partial\ell_{xe}(x_i)}{\partial f(x_i)}\right]^T\frac{\partial f(x_i)}{\partial\theta}\cdot-\frac{\partial\theta}{\partial t}\Big|_D\\
=&\frac{1}{|D|}\sum_{i\in S}\left\langle\mathbf{y}_i-p(x_i),\frac{\partial f(x_i)}{\partial t}\Big|_D\right\rangle.
\end{aligned} \tag{4.13}$$

As demonstrated in Eq. (4.10) and Eq (4.13), we can derive a score function for data selection

from the dynamics optimization formulation of curriculum learning. By selecting the samples with the highest score for training, the training dynamics can be optimized to achieve the greatest progress. We provide a detailed study of the score functions in Section 8.2.

Chapter 5

MODEL-DATA MATCHING CURRICULUM FOR DIVERSE ENSEMBLE LEARNING

In this chapter, we study how to efficiently train a diverse but complementary ensemble of models by dynamically adjusting the training data assigned to each model according to (1) their expertise evolving through the course of training; (2) intra-model diversity, i.e., the diversity of training data assigned to each model; and (3) inter-model diversity, i.e., the diversity of training data assigned to different models. This problem extends the previously studied data selection curriculum for training a single model to a data-model matching (or marriage) curriculum for training multiple models. Specifically, we formulate such a data-model marriage problem for each learning stage as a generalized bipartite matching, represented as submodular maximization subject to two matroid constraints. In SectionWe then develop a curriculum, "Diverse Ensemble Evolution (DivE^2)", which solves a sequence of continuous-combinatorial optimizations with slowly varying objectives and constraints. The combinatorial part handles the data-model marriage while the continuous part updates model parameters based on the assignments.

5.1 Learning a Diverse Ensemble of Expert Models

Ensemble methods [28, 185, 86, 29] are simple and powerful machine learning approaches to obtain improved performance by aggregating predictions (e.g., majority voting or weighted averaging) over multiple models. Over the past few decades, they have been widely applied, consistently yielding good results. For neural networks (NN) in particular, ensemble methods have shown their utility from the early 1980s [257, 78, 115, 33] to recent times [156, 77, 213, 58]. State-of-the-art

results on many contemporary competitions/benchmarks are achieved via ensembles of deep neural networks (DNNs), e.g., ImageNet [51], SQuAD [175], and the Kaggle competitions (https://www.kaggle.com/). In addition to boosting state-of-the-art performance of collections of large models, ensembles of small and weak models can achieve performance comparable to much larger individual models, and this can be useful when machine resources are limited. Inference over an ensemble of models, moreover, can be easily parallelized even on a distributed machine.

A key reason for the success of ensemble methods is that the diversity among different models can reduce the variance of the combined predictions and improve generalization. Intuitively, diverse models tend to make mistakes on different samples in different ways (e.g., assigning largest probability to different wrong classes), so during majority voting or averaging, those different mistakes cancel each other out and the correct predictions can prevail. As neural networks grow larger in size and intricacy, their variance correspondingly increases, offering opportunity for reduction by a diverse ensemble of such networks.

Randomization is a widely-used technique to produce diverse ensembles. Classical ensemble methods such as random initialization [51, 204], random forests [86, 29] and Bagging [28, 56], encourage diversity by randomly initializing starting points/subspaces or resampling the training set for different models. Ensemble-like methods for DNNs, e.g., dropout [198] and swapout [191], implicitly train multiple diverse models by randomly dropping hidden units out during the training of a single model. Diversity can also be promoted by sequentially training multiple models to encourage a difference between the current and previously trained models, such as Boosting [185, 65, 156] and snapshot ensembles [88]. Such sequential methods, however, are hard to parallelize and can lead to long training times when applied to neural networks.

Sequential methods have also been designed for training a diverse ensemble but with the limitation on parallelization. The most famous one is Boosting [185, 65, 156], which trains a sequence of models one after another, and the sample weights are computed based on the

performance of previous models (worse samples get larger weights). It encourages diversity between consecutive models by assigning different weights to same samples. Recently, snapshot ensembling [88] proposes to use a cyclic learning rate to sequentially achieve multiple local minima, which are then used to compose an ensemble. It reduces the learning rate to converge to a local minima, then uses a large learning rate to escape from it, and reduce it again to converge to the next local minima. With a sufficiently large learning rate at the beginning of each cycle, the local minima are expected to be diverse.

Very recently, sparsely gated mixture of experts [190] is proposed to train a gating network to select a sparse (e.g., tens out of thousands) subset of models from a large ensemble. It achieves compelling performance on NLP tasks, but requires expensive computations due to the end-to-end training and the large number of models. Most other ensemble methods mainly focus on using an ensemble for inference rather than training it, e.g., static model selection [33, 167, 257], dynamic classifier selection (DCS) [34, 151, 262, 44], model compression [31], knowledge distillation [84], etc. More details of the related works and comparisons can be found in Section 5.3.

Though more adaptive than randomization based methods, a potential problem of above methods is that they are not friendly to cold start, i.e., if a new member needs to join in the ensemble, all the members in it should be re-trained from very beginning.

Despite the consensus that diversity is essential to ensemble training, there is little work explicitly encouraging and controlling diversity during ensemble model training. Most previous methods encourage diversity only implicitly, and are incapable of adjusting the amount of diversity precisely based on criterion determined during different learning stages, nor do they have an explicit diversity representation. Some methods implicitly encourage diversity during training, but they rely on learning rate scheduling (e.g., snapshot ensembles [88]) or end-to-end training of an additive combination of models (e.g., mixture of experts [94, 104, 190]) to promote diversity, which is hard to control and interpret.

Moreover, many existing ensemble training methods train all models in the ensemble on all samples in the training set by repeatedly iterating through it, so the training cost increases linearly with the number of models and number of samples. In such case, each model might waste much of its time on a large number of redundant or irrelevant samples that have already been learnt, and that might contribute nearly zero-valued gradients. The performance of an ensemble on each sample only depends on whether a subset of models (e.g., half for majority voting) makes a correct prediction, so it should be unnecessary to train each model on every sample.

5.2 Motivations of Diverse Ensemble Evolution

In this chapter, we aim to achieve an ensemble of models using explicitly encouraged diversity and focused expertise, i.e., each model is an expert in a sufficiently large local-region of the data space, and all the models together cover the entire space. We propose an efficient meta-algorithm "diverse ensemble evolution (DivE^2)", that "evolves" the ensemble adaptively by changing over stages both the diversity encouragement and each model's expertise, and this is based on information available during ensemble training. It does this encouraging both intra- and inter-model diversity. Each training stage is formulated as a hybrid continuous-combinatorial optimization. The combinatorial part solves a data-model-marriage assignment via submodular generalized bipartite matchings; the algorithm explicitly controls the diversity of the ensemble and the expertise of each model by assigning different subsets of the training data to different models. The continuous part trains each model's parameters using the assigned subset of data. At each stage, all the models may be updated in parallel after receiving their assigned data.

A similar approach to encourage inter-model diversity was used in[43] but there diversity of different models is achieved by encouraging diverse subsets of features and the goal was to cluster the features into potentially overlapping groups; here we are encouraging diverse subsets of samples to be assigned to models during the training process and we are matching data samples to models.

Over different stages, the data-model marriage assignments adaptively change over time as the learners get better. The schedule of assignments, and the diversity encouragement parameters, we use is inspired by how a collection of humans learn over time. Early learners (e.g., young children) tend not to specialize as much as they do later in life and, for the most part, learn a similar concepts about the world. The early learner, however, should not learn too redundant a set of concepts as that can lead to fragility later on. Moreover, for social robustness, the community of learners should exhibit diversity. Both of these forms of diversity can be encouraged during early stages of learning. As the learning proceeds (e.g., young adults), it is beneficial to allow them to specialize and develop expertise on those concepts they exhibit an affinity to. At the same time, we must ensure a diversity of concepts amongst the learners to ensure all concepts are covered.

5.3 Related Work

5.3.1 Three mostly used classical ensemble methods

Bagging [28]: bagging samples different training sets for different models before any training starts, and train all models in parallel, finally average all models' outputs as prediction. It does not adapt with the training process, i.e., the assignment of training data does not depend the performance of any model at any training stage. Multiple models can be trained in parallel so bagging is potentially applicable to deep neural nets, but might perform worse than simple average ensemble or dropout [125].

Boosting [185, 65, 156]: train a sequence of models one after another, and the weight of each training sample to train the next model depends on its classification accuracy achieved on previously trained models. It is adaptive and can build a strong ensemble model from multiple weak learners. However, it is not practical in training deep neural nets that usually require a long training time, because each model needs to wait the previous one to converge. In addition, boosting cannot adaptively adjust the training set during the training process of each model. The data assignment

happens before any training begins.

Mixture of Experts (MoE) [94, 104]: they use a gating network to select a subset of models (experts) for each given sample. Because the gating network connects all the models together and forms a modular combination, which is usually a large neural network, end-to-end training is usually required, which is hard to parallelize and might result in expensive computations and heavy memory load. DivE^2 has similar idea of training experts, but is different from MoE methods in that 1) DivE^2 explicitly promotes diverse and complementary expertise on different experts; and 2) DivE^2 does not require end-to-end training (the gating network needs re-training if we remove or add models to the ensemble) and is able to train models in parallel, because each model is independently updated based on the assigned data in each learning stage.

5.3.2 Two mostly used ensemble methods for deep neural nets

Simple average might be the most widely used ensemble methods especially for deep neural nets. It trains different models on the same training set but initialize them randomly (and thereby promote diversity implicitly) at the beginning of optimization. It is simple to use but the diversity cannot be explicitly enforced. Moreover, it trains each model on the whole training set independently, so the training costs increase linearly with the number of models m.

Dropout [198]/Swapout [191]: implicitly gain an ensemble by randomly killing a portion of hidden nodes (i.e., set their outputs to be zeros) or skipping over layers (i.e., layer dropout). They implicitly average multiple models with different structures but with shared weights. They are different from explicitly training multiple models and explicitly enforcing the diversity between them. They can always be combined with other ensemble models including ours to further improve generation performance (in this case each model in the ensemble is implicitly an ensemble of models with different structures), and are orthogonal to methods explicitly training multiple models. ResNet [81] can also been explained as an ensemble of shallow models due to its shortcut link

between nonconsecutive layers [219].

5.3.3 Two recently proposed ensemble methods for deep neural nets

Snapshot ensemble [88]: by using a cyclic learning rate scheduling, it can quickly converge to a local minimum, and escape from it by an increasing learning rate and then converge to the next local minimum. By repeating this process for several times, it can achieve multiple local minimum models. The final ensemble is composed of the last several local minimum models. They can also be easily combined with other ensemble methods. One possible disadvantage of snapshot ensemble is that the computational cost to achieve so many local minimums (note the number could be much larger than the number of models used to compose the ensemble) can be very expensive, because it needs to sequentially get local minimum models one after another.

Sparsely gated mixture of experts [190]: it uses a parameterized gate to combine the outputs of all the models, and the gate is designed to only assign nonzero weights to a small number of models. This has been shown to be effective for some NLP tasks. The sparsity is helpful to develop diverse expertise on different models, but can easily cause imbalance loading problem in practice, as the extremely sparse weights given by the gate may always assign most data to few models. In addition, it needs to train thousands of models together with the gating network as a huge neural net in end-to-end manner, so the computational costs are very expensive, and it is not easy to synchronize the training process and accelerate it in parallel.

5.3.4 Other Related Work

Model compression [31] or knowledge distillation [84] learns a single small model to imitate an ensemble of models, so the aggregation requires much less computation and memory. These methods mainly focus on improving the aggregation efficiency, and can also be applied to the diverse ensemble achieved by DivE^2.

5.4 Diverse Ensemble Evolution: Formulation

In this section, we start from a novel optimization formulation of ensemble training in Section 5.4.1, which combines the data assignment with model updating and lets them interact with each other on the fly during training. We then introduce two matroid constraints corresponding to two learning modes, i.e., "model selecting samples" and "sample selecting models" in Section 5.4.1, and two diversity regularizations encouraging inter-model and intra-model diversities in Section 5.4.2. By equipping the optimization with these constraints and regularizations, we can explicitly and adaptively control the evolution process of expertise on all models. In Section 14.1, we show how to approximately solve this challenging continuous-combinatorial optimization by interplay between convex/non-convex optimizer and submodular maximization, with the latter detailed in Section 5.4.3. In Section 14.3, we develop a curriculum as a sequence of the optimization problems with smoothly changing constraints and regularization weights that can result in a diverse ensemble with complementary expertise. DivE^2 solves them one after another by initializing each with the solution to the previous one as a warm start, and finally produces the desirable diverse ensemble.

5.4.1 Data-Model Marriage

An ensemble of models can make an accurate prediction on a sample without requiring each model making accurate predictions on that sample [125, 73, 126]. Rather, it requires a subset of models to produce accurate predictions, and the remainder may err in different ways. Hence, rather than training each model on the entire training set, we may in theory assign a data subset to each model. Then, each sample is learned by a subset of models, and different models are trained on different subsets thereby avoiding common mistakes across models.

Consider a weighted bipartite graph (see Fig. 5.1), with the set of $n = |V|$ training samples V on the left side, the set of $m = |U|$ models U on the right side, and edges $E \triangleq \{(v_j, u_i)|v_j \in V, u_i \in U\}$ connecting all sample-model pairs with edge weights defined by the loss $\ell(v_j; w_i)$ of sample

$v_j = (x_j, y_j)$ (where x_j is the features and y_j is the label(s)) on model u_i (where model u_i is parameterized by w_i). We wish to marry samples with models by selecting a subset of edges having overall small loss. We can express this as follows:

$$\max_{\{w_i\}_{i=1}^m} \max_{A \subseteq E} \sum_{(v_j,u_i)\in A} (\beta - \ell(v_j; w_i)), \tag{5.1}$$

where $\beta - \ell(v_j; w_i)$ translates loss to reward (or accuracy), and β is a constant larger than any per-sample loss on any model, i.e., $\beta \geq \ell(v_j; w_i), \forall i, j$ [*].

With no constraints, all edges are selected thus requiring all models to learn all samples. As mentioned above, for ensembles, every sample need only be learned by a few models, and thus, for any sample v, we may wish to limit the number of incident edges selected to be no greater than k. This can be achieved using partition matroid $\mathcal{M}_V = (E, \mathcal{I}_V)$, where $\mathcal{I}_V = (I_1, I_2, \ldots, I_n)$ and $I_i \subseteq E$. $\mathcal{I}_V$ contains all subsets of E where no sample is incident to more than k edges in any subset, i.e. $\mathcal{I}_V = \{A \subseteq E : |A \cap \delta(v)| \leq k, \forall v \in V\}$, where $\delta(v) \subseteq E$ is the set of edges incident to v (likewise for $\delta(u), u \in U$). Therefore, as long as a selected subset $A \subseteq E$ satisfies the constraint ($A \in \mathcal{I}_V$), every sample is selected by at most k models.

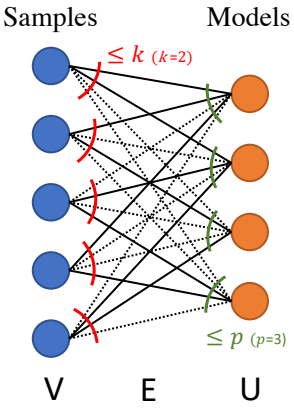

Figure 5.1: Data-Model Marriage as Bipartite Matching.

With only the constraint $A \in \mathcal{I}_v$, different models can be assigned dramatically differently sized data subsets. In the extremely unbalanced case, k models might get all the training data, while the other models get no data at all. This is obviously undesirable because k models will learn from the same data (no diversity and no specialized and complementary expertise), while the others models

[*]Although in theory the loss can be arbitrarily large, in practice, it is usually forced to be upper bounded by a constant for a stable gradient, e.g., a small σ used in $-\log(p_i + \sigma) \leq -\log(\sigma)$ when computing cross entropy loss. Gradient clipping widely used in training neural nets also avoids arbitrarily large loss.

learn nothing. Running time also is not improved since training time is linear in the size of the largest assigned data set, which is all of the data in this case. We therefore introduce a second partition matroid constraint $\mathscr{M}_U = (E, \mathscr{I}_U)$, which limits to p the number of samples selected by each model. Specifically, $\mathscr{I}_u = \{A \subseteq E : |A \cap \delta(u)| \leq p, \forall u \in U\}$. Eq. (5.1) then becomes:

$$\max_{\{w_i\}_{i=1}^m} \max_{A \subseteq E, A \in \mathscr{I}_v \cap \mathscr{I}_u} \sum_{(v_j, u_i) \in A} (\beta - \ell(v_j; w_i)). \tag{5.2}$$

The interplay between the two constraints (i.e., $\mathscr{I}_v$: k models per sample, and $\mathscr{I}_u$: p samples per model) is important to our later design of a curriculum that leads to a diverse and complementary ensemble. When $mp < nk$, $\mathscr{I}_u$ tends to saturate (i.e., $|A \cap \delta(u)| = p, \forall u \in U$) earlier than $\mathscr{I}_v$. Hence, each model generally has the opportunity to select the top-p easiest samples (i.e., those having the smallest loss) for itself. We call this the "model selecting sample" (or early learning) phase, where each model quickly learns to perform well on a subset of data. On the other hand, when $mp > nk$, $\mathscr{I}_v$ tends to saturate earlier (i.e., $|A \cap \delta(v)| = k, \forall v \in V$), and each sample generally has the opportunity to select the best-k models for itself. We call this the "sample selecting model" (or later learning) phase, where models may develop complementary expertise so that together they can perform accurately over the entire data space. We give conditions on which phase dominates in Lemma 3.

5.4.2 Inter-model & Intra-model Diversity

To encourage different multiple models to gain different proficiencies, the subsets of training data assigned to different models should be diverse. The two constraints introduced above are helpful to encourage diversity to a certain extent when p and k are small. For example, when $k = 1$ and $p \leq n/m$, no pairs of models share any training sample. Different training samples, however, can still be similar and thus redundant, and in this case the above approach might not

encourage diversity when p or k is large. Therefore, we incorporate during training an explicit inter-model diversity term $F_{inter}(A) \triangleq \sum_{i,j\in[m],i<j} F\big((\delta(u_i) \cup \delta(u_j)) \cap A\big)$ and add it to the objective function in Eq. (5.2). This discourages model pairs from becoming too similar by discouraging their being assigned similar data. The set function $F : 2^E \to \mathbb{R}_+$ is chosen from the large expressive family of submodular functions, which naturally measure the diversity of a set of items [66]. A submodular function satisfies the diminishing return property: given a finite ground set V, any $A \subseteq B \subseteq V$ and an element $v \notin B, v \in V$, we have $F(\{v\} \cup A) - F(A) \geq F(\{v\} \cup B) - F(B)$. Submodular functions have been applied to a variety of diversity-driven tasks to achieve good results [136, 134, 13, 171, 69, 64].

Another issue of Eq. (5.2) is that each model might select easy but redundant samples when constraint $\mathcal{F}_u$ dominates (the model-selecting-sample phase). This is problematic as each model might quickly focus on a small group of easy samples, and may overfit to such small region in the data space. We therefore introduce another set function $F_{intra}(A) = \sum_{i\in[m]} F'(\delta(u_i) \cap A)$ to promote the diversity of samples assigned to each model. Similar to F, we also choose F' to be a submodular function. The optimization procedure now becomes:

$$\max_{W} \max_{A\subseteq E, A\in\mathcal{F}_v\cap\mathcal{F}_u} G(A, W) \triangleq \sum_{(v_j,u_i)\in A} (\beta - \ell(v_j; w_i)) + \gamma F_{inter}(A) + \lambda F_{intra}(A), \tag{5.3}$$

where γ and λ are two non-negative weights to control the trade-offs between the reward term and the diversity terms, and we denote $W \triangleq \{w_i\}_{i=1}^m$ for simplicity. By optimizing the objective $G(A, W)$, we explicitly encourage model diversity in the ensemble, while ensuring every sample gets learned by k models so that the ensemble can generate correct predictions. A form of this objective has been called "submodular generalized matchings" [10] where it was used to associate peptides and spectra.

5.4.3 *Data Assignment as a Generalized Bipartite Matching Problem*

The combinatorial optimization problem in Eq. (5.3) is a generalized bipartite matching problem [135] with monotone submodular evaluations and two matroid constraints, which is a special case of monotone submodular maximization with p-matroid constraint ($p = 2$). simple greedy algorithm can yield an approximation factor of $\alpha = 1/p+1$ [63]. This result can be further improved when the objective $G(\cdot, W)$ is close to modular. Specifically, α becomes $\alpha = 1/p+\kappa_G$ [41], which depends on the curvature $\kappa_G \in [0, 1]$ defined as

$$\kappa_G \triangleq 1 - \min_{j \in V} \frac{G(j|V \backslash j)}{G(j)}. \tag{5.4}$$

When $\kappa_G = 0$, $G(\cdot, W)$ is modular, and when $\kappa_G = 1$, $G(\cdot, W)$ is fully curved and the above bound recovers $\alpha = 1/p+1$. The objective $G(\cdot, W)$ in Eq. (5.3) is weighted sum of a modular function and two submodular functions. It becomes closer to modular as the weights λ and γ for the two submodular functions decrease, and κ_G decreases accordingly. We therefore have the following Lemma:

Lemma 1. *Let $G(A) = M(A) + \lambda F(A)$ where $F(\cdot)$ is a monotone non-decreasing submodular function with curvature κ_F, $M(\cdot)$ is a non-negative modular function, and $\lambda \geq 0$. Then $\kappa_G \leq \frac{\kappa_F}{c/\lambda+1}$ where $c = \min_{j \in V} M(j)/F(j)$.*

Proof. We have

$$\begin{aligned}
\kappa_G &= 1 - \min_{j \in V} \frac{M(j) + \lambda F(j|V \setminus j)}{M(j) + \lambda F(j)} = \lambda \cdot \max_{j \in V} \frac{F(j) - F(j|V \setminus j)}{M(j) + \lambda F(j)} \\
&= \lambda \cdot \max_{j \in V} \frac{1 - \frac{F(j|V \backslash j)}{F(j)}}{\frac{M(j)}{F(j)} + \lambda} \leq \frac{\lambda \cdot \kappa_F}{\min_{j \in V} \frac{M(j)}{F(j)} + \lambda} = \frac{\kappa_F}{c/\lambda + 1}
\end{aligned}$$

Where $c \triangleq \min_{j \in V} \frac{M(j)}{F(j)}$. □

In DivE^2, we apply the fast greedy procedure mentioned earlier [158, 152, 153] to the data assignment problem. It secures an approximation factor $\alpha = 1/2+\kappa_G$, but might perform much better in practice.

Chapter 6

OTHER FORMULATIONS

In this chapter, we will briefly discuss other possible formulations for curriculum learning, which are worth exploring in the future works.

6.1 Online Learning and Multi-Armed Bandit

In curriculum learning, at every step, the algorithm selects a training sample (or more generally, every candidate for a learning component) and then receives a reward, e.g., the loss decrement on a validation set or on the whole training set, or a score reflecting the contribution of the selected sample. By treating each sample as an arm and the curriculum pulls an arm per step, the problem can be formulated as a online learning problem whose goal is to maximize the sum of the rewards over time. Here, the regret can be defined as the difference between the accumulated rewards achieved by the curriculum and that achieved by an optimal curriculum strategy. Hence, it is possible to formulate the problem as a multi-armed bandit problem, for which a rich class of off-the-shelf algorithms can be applied under different assumptions. By performing an exploration-exploitation trade-off, we can evaluate the utility of each sample through early exploration and build an effective curriculum by exploiting the evaluated utility values.

Stochastic bandits and The Upper Confidence Bound (UCB) Algorithms. If we assume that each sample-i is associated with a fixed mean value μ_i over the course of training and the reward R_{t,i_t} observed in every step-t is a noisy observation of the value μ_{i_t} for the pulled arm i_t, we can formulate the curriculum learning as "optimization in the face of uncertainty" and apply the UCB

algorithm to maximize the expected cumulative reward over a finite horizon T steps in total, i.e.,

$$\mathbb{E}\left[\sum_{t=1}^{T} R_{t,i_t}\right]. \tag{6.1}$$

This is equal to minimizing the regret over the T step when compared with a competing baseline. Unlike conventional definitions of regret, which adopts the best arm $\mathrm{argmax}_{i\in[n]}\,\mu_i$ as the baseline, an eligible baseline for curriculum learning has to be a subset of samples $A \subseteq [n]$ with $|A| = m$. For example, when the rewards are nonnegative, we can replace the constraint $|A| = m$ with $|A| \leq m$ and the regret can be defined as

$$R(T) = T \max_{A\subseteq[n],|A|\leq m} \frac{1}{m}\sum_{i\in A}\mu_i - \mathbb{E}\left[\sum_{t=1}^{T} R_{t,i_t}\right]. \tag{6.2}$$

The constraint for A is not limited to the cardinality constraint above but can be more complicated or general set systems. For example, to enforce the balance of all the c classes in the selected subset A, we can consider a partition matroid constraint $|A \cap C_i| \leq \frac{m}{c}$, where $C_i \subseteq [n]$ is the set of all samples belonging to class-i.

Stochastic bandits problems can usually be addressed by UCB alogrithms. Specifically, an UCB-type algorithm keeps updating an upper bound for μ_i based on all the received rewards in hindsight steps when pulling the arm of sample-i. The upper bound adds a deviation bound to the empirical average estimation for $\hat{\mu}_i$. By pulling the arm with the highest upper bound in every step, UCB can balance between selecting the arm with the highest estimated value $\hat{\mu}_i$ (exploitation) and the arms pulled for fewer times (exploration).

Adversarial bandits and Exp3 Algorithms. In a more challenging setting, i.e., when the reward $R_{t,i}$ for each arm-i in every step-t is chosen by an adversary in advance before playing, a more randomized solution is preferred since the adversary can easily turn the highest valued arm into the

worst one by controlling the reward. Specifically, instead of pulling the best arm in hindsight, the algorithm pulls an arm drawn from a distribution p_t at step-t. A representative algorithm to address adversarial bandits is Exp3, which aims at maximizing the expected cumulative reward

$$\mathbb{E}\left[\sum_{t=1}^{T}\langle p_t, R_t\rangle\right]. \tag{6.3}$$

The probability p_t in Exp3 aims at approximating the expert setting of the multiplicative weight algorithm or Exponentiated Gradient (EG), which assumes that the whole vector R_t is observable in every step, i.e.,

$$p_t[i] \propto \exp\left(\sum_{\tau=1}^{t-1}\eta R_{t,i}\right). \tag{6.4}$$

Similar to the stochastic bandits for curriculum learning, the corresponding regret for curriculum learning in this case can be defined as

$$R(T) = \max_{A\subseteq[n],|A|\leq m}\sum_{t=1}^{T}\frac{1}{m}\sum_{i\in A}R_{t,i} - \mathbb{E}\left[\sum_{t=1}^{T}\langle p_t, R_t\rangle\right]. \tag{6.5}$$

Exp3 only observes R_{t,i_t} for the pulled arm i_t only and replaces $R_{t,i}$ with an unbiased estimate $\hat{R}_{t,i}$, which is $R_{t,i_t}/p_{t,i_t}$ for $i = i_t$ and 0 otherwise.

Bayesian Bandits and Thompson sampling. In contrast to estimating the mean value of each arm and pulling the highest valued arm, Thompson sampling provides a Bayesian solution that estimates the whole distribution (not just the mean) of the reward for each arm: it samples a value from the posterior distribution for each arm in every step and pull the arm with the highest sampled value. Starting from a prior distribution (e.g., uniform distribution), Thompson sampling keeps updating the posterior distribution for each arm using the observed rewards. Therefore, it can balance between the exploration and exploitation: on the one hand, the arms pulled for fewer times

will have more uniform distributions and thus have large values by chance; on the other hand, the arms pulled for many times with higher observed rewards will have more concentrated distributions at high values. By replacing the parametric forms of prior and posterior distributions in Thompson sampling with nonparametric models such as Gaussian process, we can also consider Bayesian optimization that aims to maximize a black-box function mapping from arms/actions to rewards.

For curriculum learning, multi-armed bandit (MAB) algorithms are capable to provide an accurate estimation of the scores of samples with efficient exploration-exploitation trade-off, so the curriculum only needs a short earlier exploration stage of training to allocate the most informative samples for the future training and thus accelerate the convergence of later stages. That being said, there are two major limitations when formulating curriculum learning as an MAB problem: (1) MAB usually assumes that the observed reward is drawn from a stationary distribution or pre-determined by an adversary, which does not hold for curriculum learning, because the utility or importance of each sample varies across different training stages and depends on the number of times that the sample has been used for training; (2) MAB algorithms only pull one single arm in every step. Although it is not invalid to select one training sample for each step, most curriculum learning algorithms, when applied in practice, usually select one or multiple mini-batches of samples for training in every learning stage. In addition, to achieve a reasonable regret bound, MAB algorithms cannot tolerate a large number of arms but curriculum learning usually requires to select samples from a large-scale dataset, which could be in the tens of millions or billions or larger.

In order to address the first limitation, we can downweigh the earlier stage feedback in the decision making of later steps. In particular, we can study the non-stationary or multi-stage variants of MAB algorithms, which is built upon the assumption that the true reward distribution can change between two stages. For example, we can develop estimators of the mean value in UCB, the reward distribution in Thompson sampling, or the importance weights in Exp3, which downweigh or drop the older feedback or observations and focus on more recent ones [111, 67, 79].

To address the second limitation, we can extend the previous setting from pulling a single arm per step to the setting of combinatorial bandits that can pull a subset of arms per step.

Stochastic Combinatorial Bandits In combinatorial bandits, the player in each step selects a super-arm $A_t \subseteq [n]$ as a subset composed of m samples. The feedback can be either semi-bandit (i.e., $R_{t,i}$ for every $i \in A_t$ is revealed) or the more challenging bandit feedback (e.g., only $sum_{i \in A_t} R_{t,i}$ is revealed). The goal of a combinatorial bandit algorithm is to maximize the expected cumulative reward of a sequence of subsets $(A_1, A_2, \cdots, A_t)$,

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{i \in A_t} R_{t,i}\right], \tag{6.6}$$

or minimize the regret

$$R(T) = T \max_{A \subseteq [n], |A| \leq m} \sum_{i \in A} \mu_i - \mathbb{E}\left[\sum_{t=1}^{T}\sum_{i \in A_t} R_{t,i}\right]. \tag{6.7}$$

This can be a challenging task since the number of total arms can exponentially increase with the number of total samples. Fortunately, by exploiting the linear (or modular) structure of the reward, it is still possible to achieve an efficient algorithm. However, the above bandit feedback does not consider the dependency among samples selected into A_t. More general settings can assume nonlinear reward $F(\{R_{t,i}\}_{i \in A_t})$ or non-modular reward $F(A_t)$ (e.g., $F(A_t)$ can be a submodular or supermodular function). In order to address these settings, exploitation of the nonlinear or non-modular structure is necessary to reduce the exploration cost of the space of arms/actions.

Adversarial Combinatorial Bandits Similar to stochastic combinatorial bandits for curriculum learning, we can also extend the adversarial bandits from pulling a single arm to pulling a subset of arms $A_t \subseteq [n]$ in every step. The resulted adversarial combinatorial bandit problem aims to

minimize the regret

$$R(T) = \max_{A \subseteq [n], |A| \leq m} \sum_{t=1}^{T} R_{t,A} - \mathbb{E}\left[\sum_{t=1}^{T} \sum_{A \subseteq [n], |A| \leq m} p_t[A_t] R_{t,A_t}\right], \tag{6.8}$$

where $R_{t,A}$ is the reward for selecting A and it can be either the a linear/modular function $R_{t,A} = \sum_{i \in A} R_{t,i}$ or a nonlinear/non-modular function that captures the dependencies among selected samples. The probability $p_t[A_t]$ of selecting a subset A_t at step-t can be modeled by a log-modular/submodular/supermodular distribution $p_t[A_t] \propto \exp(F(A_t))$ when $F(A_t)$ is modular/submodular/supermodular, or by a distribution produced by submodular/supermodular point process (SPP) [92]. Moreover, similar to stochastic combinatorial bandits, we can consider different constraints defining the set system for A and exploit their structures to facilitate the exploration.

6.2 Reinforcement Learning

Reinforcement learning (RL) extends the multi-armed bandits to non-stationary and the general Markov decision processes, where the value for each arm/action varies over time steps and depends on the the state of every step. In curriculum learning, the importance of each sample also changes when the capability and performance of the learner (i.e., the model) change during the course of training. Hence, formulating curriculum learning as an RL problem is able to capture the dynamics and adaptivity of curricula. However, it also raises new challenges because the optimization in RL is notoriously difficult and its sample complexity is high, especially for a complicated state space and action space (which is the case for curriculum learning).

We can describe curriculum learning as a Markov decision process (MDP) $\{\mathcal{S}, \mathcal{A}, p, r, \gamma\}$, where the state space $\mathcal{S}$ covering all the possible phases for training a single model [*], $\mathcal{A}$ is the

[*]In the worst and the most general case, it can be the space of all feasible models since an arbitrary training process may reach any of them in every step. However, the space might be reduced to a smaller one for a specific class of initialization, a particular optimizer, and limited learning rates and steps.

action space of the curriculum (e.g., $[0,1]^n$ for actions as the weights of the n training samples), $p(s'|s,a) \triangleq \Pr(s_{t+1}=s'|s_t=s,a_t=a)$ is the transition probability for the model from state s to s' after taking action a and is defined by the optimizer, $r(s,a):\mathcal{S}\times\mathcal{A}\times\mathcal{G}\mapsto\mathbb{R}$ is a reward function (e.g., the improvement on the validation loss), and $\gamma\in[0,1]$ is a discount factor. The goal of curriculum learning as an RL problem is to learn a curriculum policy $\pi(a|s):\mathcal{S}\times\mathcal{G}\mapsto\mathcal{A}$ that outputs an action a (or probabilities $\Pr(a|s)$ over actions $a\in\mathcal{A}$) given a model state s. The curriculum policy uses $\pi(a|s)$ to interact with the Markov decision process (MDP).

RL policy usually need to be trained for multiple episodes. In each episode, the agent starts from an initial model $s_0\sim p_0(s)$. In every time step t with model s_t, the curriculum takes an action $a_t=\pi(a|s_t)$ (deterministic) or $a_t\sim\pi(a|s_t)$ (stochastic) by selecting or sampling a subset of training data, receives a reward $r(s_t,a_t)$, and the model is updated to a new state $s_{t+1}\sim p(s'|s_t,a_t)$. We define the discounted return as $R_t=\sum_{i=t}^{T}\gamma^{i-t}r(s_t,a_t|g)$. RL aims to learn a policy π maximizing the expected return $\mathbb{E}_{(s_0)}[\mathbb{E}_\pi(R_0)]$. Define the action-value function $Q(s,a)\triangleq\mathbb{E}(R_t|s_t=s,a_t=a)$, the optimal policy π^* achieves the maximal $Q(s,a)$ for any feasible (s,a), i.e., π^* results in $Q^*(s,a)\triangleq\max_\pi Q(s,a)$ that satisfies the Bellman equation:

$$Q^*(s_t,a_t)=r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}}[V^*(s_{t+1})], \tag{6.9}$$

Define the value function $V(s)\triangleq\mathbb{E}(R_t|s_t=s)=\mathbb{E}_{a\sim\pi}[Q(s,a)]=\sum_{a\in\mathcal{A}}\pi(a|s)Q(s,a)$, the optimal $V^*(s)\triangleq\max_\pi V(s)=\max_{a\in\mathcal{A}}Q^*(s,a)$. Due to these definitions, the optimal policy π^* can be extracted from V^* or Q^*.

Directly maximizing the expected return or V w.r.t. π results in the vanilla policy gradient method [202], which can be sample-inefficient and suffers from the high variance of R_t. Actor-critic methods [203] additionally learns a model for V or Q as a "critic" to the "actor" π, which can perform as a baseline in order to effectively reduce the variance. According to the Bellman equation

in Eq. (6.9), the optimization of V or Q aims to minimize the Bellman residual

$$J_{Q^\pi} = \mathbb{E}_{(s_t,a_t)}\left[Q^\pi(s_t,a_t) - r(s_t,a_t) - \gamma \mathbb{E}_{s_{t+1}}[V(s_{t+1})]\right]^2, \tag{6.10}$$

Given the critic Q, maximizing the expected return w.r.t. π reduces to minimizing

$$J_\pi = \mathbb{E}_{(s_t)}[-V(s)] = \mathbb{E}_{(s_t)}[\mathbb{E}_{a_t}[-Q^\pi(s,a)]]. \tag{6.11}$$

A typical actor-critic algorithm alternates between minimizing J_Q and J_π.

Although the RL formulation for curriculum learning can capture the non-stationary dynamics of the learning process, training an RL policy in such a scenario is usually challenging because the space of actions and states for curriculum learning are much more complicated than that commonly studied in RL. In order to make the RL solution more practical, it is critical to simplify the state/action space and collect dense rewards to provide informative and sufficient feedback to train the policy. Moreover, RL usually requires multiple episodes of training in the same environment, so the cost of training a curriculum policy only for a specific dataset or task could be overly expensive. Furthermore, whether the learned RL policy can generalize to training a model with different initialization/architecture/hyperparameters is still an open problem. Hence, a more practical direction is to train a curriculum policy that can be transferred or generalized to new datasets or tasks.

6.3 Meta-Learning or "Learning to Learn"

Since a primary goal of curriculum learning is to optimize the generalization performance of the model on a test set or a distribution of downstream test tasks, we can formulate curriculum learning as meta-learning [62] or "learning to learn" that optimizes a weight ν_i for each training sample-$i \in V^{train}$ such that the solution model minimizing the weighted training loss achieves the minimal

loss on an held-out validation set V^{val}. This problem can be formulated as a bi-level optimization, i.e.,

$$\begin{aligned} &\min_{\nu} \frac{1}{|V^{val}|} \sum_{j \in V^{val}} L(y_j, f(x_j, w^*)) \\ &s.t. \;\; w^* \in \operatorname*{argmin}_{w} \sum_{i \in V^{train}} \nu_i L(y_i, f(x_i, w)), \end{aligned} \tag{6.12}$$

where the constraint can be relaxed as a an output solution of running a learning algorithm that aims at minimizing the weighted training loss, e.g., k-step gradient descent or SGD. Since back-propagation through all the learning steps can be expensive on both memory and computation, the techniques proposed in the first-order approximation of meta-learning (e.g., first-order MAML or Reptile [162]) or implicit MAML [174] can be applied to develop a more practical algorithm to optimize the sample weights. In addition, we can extend the sample weights to a trainable model generating the weight for any given sample so the curriculum can be generalized to unseen training data.

Meta-learning formulation can provide a principal objective for curriculum learning and thus has the potential to inspire more effective curricula. However, it requires multiple training trajectories (i.e., each solving an optimization in the constraint) to train the weights, which cost can be expensive and thus weakens the advantage of curriculum learning on improving the training efficiency. Moreover, it is still an open challenge to train a meta-model that can automatically produce the weight for arbitrary training sample.

Part II

SCORING FUNCTIONS FOR CURRICULUM LEARNING

In curriculum learning, a primary challenge is to evaluate the utility or contribution of each candidate sample or task to the future learning process by a score function. Thereby, the model can mainly focus on learning the most informative samples with higher scores that potentially lead to the greatest improvement in the future training. Early works of curriculum learning [106, 12, 197] rely on human experts [19] or domain knowledge to provide the score function in order to build a curriculum. Although their effectiveness have been demonstrated to certain extent, they are not always available or accurate in practice. Moreover, the scores are non-adaptive and unaware of the training process of the model. To remove this constraint, various recent approaches choose to define a score function based on the feedback from the model itself. Self-paced learning (SPL) [116, 207, 201, 208] chooses the curriculum based on hardness scores (e.g., per-sample loss) during training. SPL selects samples with smaller loss under a certain threshold, and gradually increases the threshold (hence the size of the selected subset also increases) over time until all the training data are selected. Moreover, self-paced curriculum learning [101] combines the human expert score and the loss-based hardness score in SPL. SPL with diversity (SPLD) [100] subtracts a group sparsity score from the objective (i.e., the loss) that SPL aim to minimize. In addition, SPLD increases the weight of the diversity score in later stages so the focus of training gradually changes from the easiest samples to diverse samples.

In this part, we will discuss different designs of the score function in curriculum learning and the principles or motivations behind them. We will start from the widely used scores based on instantaneous feedback from the model and then propose scores defined on the training dynamics.

We will analyze how these scores and the patterns of training dynamics they capture can be utilized to evaluate the importance and contributions of training samples for the future-stage learning in a curriculum.

Chapter 7

INSTANTANEOUS FEEDBACK

Instantaneous feedback from the model has been widely used in previous work as score functions for curriculum learning. It can reflect how different the candidate samples are for the most up-to-date model in the current training step. But it cannot capture how the model output changes during the course of training, i.e., the training dynamics of samples, which might contain critical information of identifying the important samples that can improve the training dynamics in the later stages.

7.1 Hardness

Probably hardness is the most widely studied score function to construct an effective curriculum. A variety of curriculum learning methods select samples in each step according to how difficult they are for the model in the current step since the hardness reflects the weakness of the model. For example, previous works [195, 172, 194, 252] use "instance hardness" defined as $1 - p_w(y_i|x_i)$, i.e., the complement of the posterior probability of label y_i given input x_i for the i^{th} sample under model w. However, different methods can differ on their preference of hard samples in different training stages. For example, self-paced learning adopts an easy-to-hard curriculum that always focuses on the easiest samples with the smallest loss values and progressively includes more samples to make the later-stage training more challenging. The training converges faster with the easiest samples only and provides a warm-up for addressing the harder ones in later stages. In contrast, a great number of ML methods in boosting and active learning always focus on the hardest samples since the model have not learned to fit them and thus they are the most informative to learn. In Chapter 2 and Chapter 8, we will introduce a novel curriculum learning method called "minimax curriculum

learning (MCL)" that also prefers more difficult examples in its data selection criterion and uses a hardness score to allocate these examples.

7.2 Subset Diversity

In curriculum learning, when selecting a small subset of samples for training, a natural goal is to keep the subset representative of the ground set of all the available training samples. Diversity of the subset in the model's latent representation space(s) can be used to measure how representative the subset is and is another type of instantaneous feedback. The latent space representations are usually compact and semantic features. Moreover, they are model-dependent so the diversity reflects the expressiveness and coverage of the model's representations.

Given a subset A, the diversity score function $F(A)$ may be chosen from the large expressive family of submodular functions, all of which are natural for measuring diversity, and all having the following diminishing returns property: given a finite ground set V, and any $A \subseteq B \subseteq V$ and a $v \notin B$,

$$F(v \cup A) - F(A) \geq F(v \cup B) - F(B). \tag{7.1}$$

This implies v is no less valuable to the smaller set A than to the larger set B. The marginal gain of v conditioned on A is denoted $f(v|A) \triangleq f(v \cup A) - f(A)$ and reflects the importance of v to A. Submodular functions [66] have been widely used for diversity models [136, 134, 13, 171, 69, 92, 25].

In minimax curriculum learning (MCL) [248] introduced in Chapter 2 and Chapter 8, we argue that the diversity of samples [230, 105, 223] is more critical in early learning since it encourages exploration, while completeness over the whole training set becomes more useful later. Although it is an instantaneous feedback metric, we empirically find that it is stable over time and does not need to be updated frequently. Actually, in various cases, the subomdular function defined on early-stage features suffice to work promisingly. Besides MCL, several curriculum learning

algorithms proposed in this thesis also include the diversity score in the curriculum for early-stage exploration. We consistently observe that the diversity can bring non-trivial improvement on sample efficiency and early-stage convergence in different experimental settings.

Chapter 8

SCORES BASED ON TRAINING DYNAMICS

A curriculum plays an important role in human learning. Given different curricula of the same training materials, students' learning efficiency and performance can vary drastically. A good teacher is able to choose the contents of the next stage of learning according to a student's past performance. Analogously, in machine learning, instead of training the model with a random sequence of data, recent work in *curriculum learning* (CL) [19, 116, 101, 248, 75] shows that manipulating the sequence of training data can improve both training efficiency and model accuracy. In each epoch, CL selects a subset of training samples based on the difficulty and/or the informativeness of each sample — this is usually measured using *instantaneous feedback* from the model (e.g., the loss), which has been introduced above. CL then uses only these samples to update the model. Inspired by human learning curricula, a schedule of training samples is constructed (e.g., from easy to hard), sometimes combining with other criteria (e.g., diversity). As exhibited in previous work, CL can help to avoid local minima, improve the training efficiency, and can lead to better generalization performance.

Instantaneous hardness, however, does not take the training history of each sample into account. When applied to deep neural network (DNNs) training, and due to the non-smooth/non-convex nature of the loss and the randomness of stochastic gradient descent (SGD), the instantaneous hardness of each sample can change dramatically between consecutive epochs, so it is not reflective of the utility of each sample in the future. This results in a large difference between training sets selected over successive epochs, leading to an inconsistency of optimization objectives and gradients, and making training less stable. Furthermore, keeping instantaneous hardness up to date requires

extra inference steps of a model over all the samples, which can be expensive for DNNs [35, 98]. Though some recent work finds that data selection within each mini-batch [103] or based on the latest evaluated (but outdated) loss [141] may still perform well, this selection can be sub-optimal and unstable.

In order to overcome the drawbacks of the instantaneous feedback, we instead investigate the patterns of training dynamics on each sample. For labeled data in supervised learning, we propose a class of training dynamics based scores called "dynamic instance hardness (DIH)" [253]. For unlabeled data in semi-supervised learning, we develop a score function called "time-consistency (TC)" [251] to select those with correct pseudo labels. For data in noisy-label learning, we combine the ideas of DIH and TC to form a composite score function that can be used to select the samples with the correct given labels or correct pseudo-labels. Though developed for different learning settings and with different motivations, these scores are based on a lazy exponential moving average (EMA) of an instantaneous feedback $a_t(i)$ collected for the subset S_t of selected training samples at each step-t, i.e., it only updates the score for every sample $iin S_t$ at step t. Specifically, the score can be defined as $\hat{a}_{t+1}(i)$ that is computed recursively as

$$\hat{a}_{t+1}(i) = \begin{cases} \gamma \times a_t(i) + (1-\gamma) \times \hat{a}_t(i) & \text{if } i \in S_t \\ \hat{a}_t(i) & \text{else ,} \end{cases} \tag{8.1}$$

We found this form of metrics captures informative patterns of training dynamics that are critical to the design of curriculum learning algorithms. In addition, it has several other advantages such as the stability over time and the computational efficiency (by only using the by-products of training on the selected samples). That being said, the class of scores based on training dynamics are not limited to the above form and it is worth exploring other metrics in the future.

8.1 Dynamic Instance Hardness (DIH)

In this section, we study the training dynamics of DNNs on individual samples from which a more accurate hardness measure can be computed that does not require extra inference and that can significantly improve performance. We study the difficulty a model has over time (i.e., training epochs) in learning each training sample. We introduce "*dynamic instance hardness* (**DIH**)" as the exponential moving average of an instantaneous hardness measure of a sample over time. We use three types of instantaneous hardness to compute DIH (fully defined later): the loss; the loss change; and the prediction flip (the 0-1 indicator of whether the prediction correctness changes) between two consecutive time steps. The first has been commonly used in CL, while the latter two capture a form of momentum of the loss/prediction.

We exploit several DIH properties that enable more effective CL approaches. Firstly, DIH can vary dramatically between different samples. Samples with smaller DIH seem to be more memorable (i.e., are retained more easily), while samples with larger DIH are harder to learn and retain. While the model is more likely to stay at a minimum of the easy samples' loss, its prediction on the hard samples is less stable under changes in optimization parameters (e.g., the learning rate). Secondly, unlike instantaneous hardness, the DIH status of a sample becomes consistent only after a few epochs. That is, a sample's DIH value converges quickly to its final relative position amongst all of the samples. For example, if a sample's DIH quickly becomes small, it stays small relative to the other samples; if it becomes large, it stays there. We can therefore accurately identify categories of hard and easy samples relatively early in the course of training. Thirdly, the DIH of each sample tends to monotonically decrease during training. This implies that the learning process strives for better local minimum for all samples, i.e., while easy samples stay easy throughout training, the hard samples also become easier the more we train on them.

8.1.1 Related Work

A special case of DIH has been studied in [212], which computes the mean of the prediction flips over all the steps after training has occurred. They show that removing samples with the smallest prediction flip average from the training set leads to less degradation of generalization performance than removing random samples. Based on this observation, they propose to train a small neural net beforehand to determine hard samples, which are then used to train a large neural net. By contrast, our study of DIH focuses on its dynamic properties **during** training, which inspires a novel curriculum learning strategy that can be applied to each step before training completes. Historical dynamics has been used to estimate prediction uncertainty over time in [35] and MentorNet [103]. However, they still rely on the instantaneous difference of a loss to its historical moving average.

The training dynamics in this section is also related to the memorization studied in [242], which considers overfitting on noisy data with random labels. We discuss this in Figure 8.6) showing that noisy data has distinctive training dynamics. Our observations also suggest that learning simple patterns [7] happens mainly from the easily memorable samples early during training. Our problem is distinct from catastrophic forgetting [177, 148, 109], which considers sequential learning of multiple tasks, where later learned tasks make the model forget what has been learned from earlier tasks. In our work, we consider single task learning.

8.1.2 Exponential Moving Average of Instantaneous Hardness

Let $a_t(i)$ be a measure of instantaneous (i.e., at time t) hardness of a sample (x_i, y_i) with feature x_i and ground truth label y_i, where i is a sample index and t is training iteration (typically, a count of mini-batches that so far have been processed). We consider three different metrics of instantaneous instance hardness in this work:

(A) Loss evaluation $\ell(y_i, F(x_i; w_t))$, where $\ell(\cdot, \cdot)$ is a standard loss function and $F(\cdot; w)$ is the model where w are the model parameters;

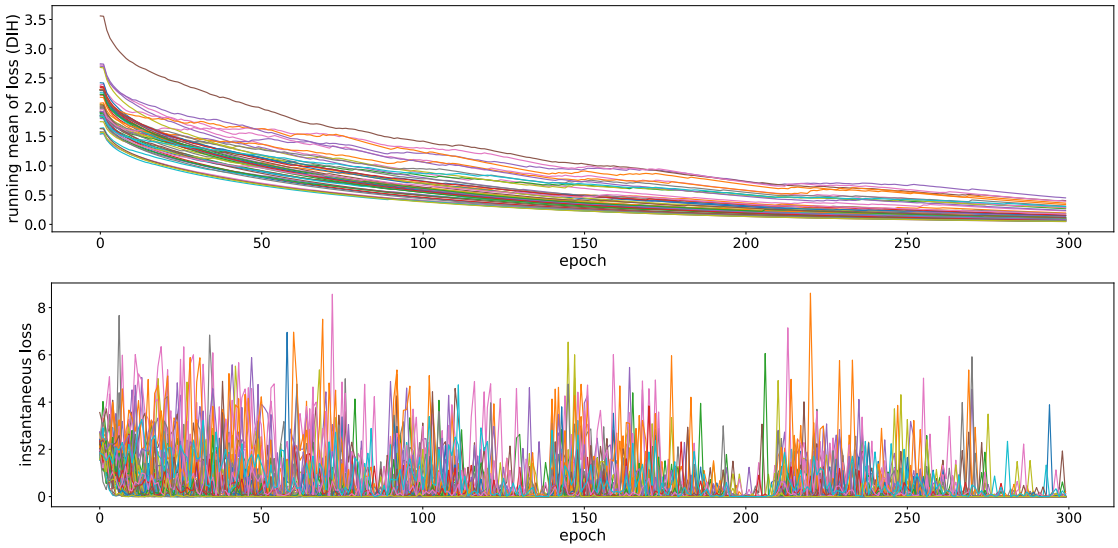

Figure 8.1: **Top:** DIH (running mean of loss) vs. **Bottom:** instantaneous loss of 50 randomly selected samples from CIFAR10 on WideResNet-28-10.

(B) Loss change $|\ell(y_i, F(x_i; w_t)) - \ell(y_i, F(x_i; w_{t-1}))|$ between two consecutive time steps;

(C) Prediction flip $|\mathbb{1}[\hat{y}_i^t = y_i] - \mathbb{1}[\hat{y}_i^{t-1} = y_i]|$, where $\hat{y}_i^t$ is the prediction of sample i in step t, e.g., $\operatorname{argmax}_j F(x_i; w_t)[j]$ for classification.

(A) corresponds closely to the "instance hardness" of [195]. However, (B) and (C) require information from previous time steps and aim to capture a form of momentum. Nevertheless, we consider (A), (B), and (C) all to be variations of instantaneous instance hardness since they use information from only a local time window around iteration t. We define dynamic instance hardness (DIH) as a running average over an instantaneous instance hardness, defined and computed recursively as

$$r_{t+1}(i) = \begin{cases} \gamma \times a_t(i) + (1-\gamma) \times r_t(i) & \text{if } i \in S_t \\ r_t(i) & \text{else ,} \end{cases} \tag{8.2}$$

where $\gamma \in [0, 1]$ is a discount factor, $S_t \subseteq V$, and $V = [n]$ is the set of all n training sample indices. S_t is the set of samples used for training at time t, e.g., a subset selected by some curriculum

learning method (or a random batch in some cases). In general, S_t should be large early in training, but as $r_t(i)$ decreases for many samples, choosing a smaller but wiser S_t will result in faster training and more accurate models. The work of [212] uses a special case of DIH at $t = T$ (T is the total number of training steps) in Eq. (8.2) with $\gamma = 1/t+1$, $S_t = V$, and $a_t(i)$ being prediction flips (case (C)).

8.1.3 Empirical Evidence

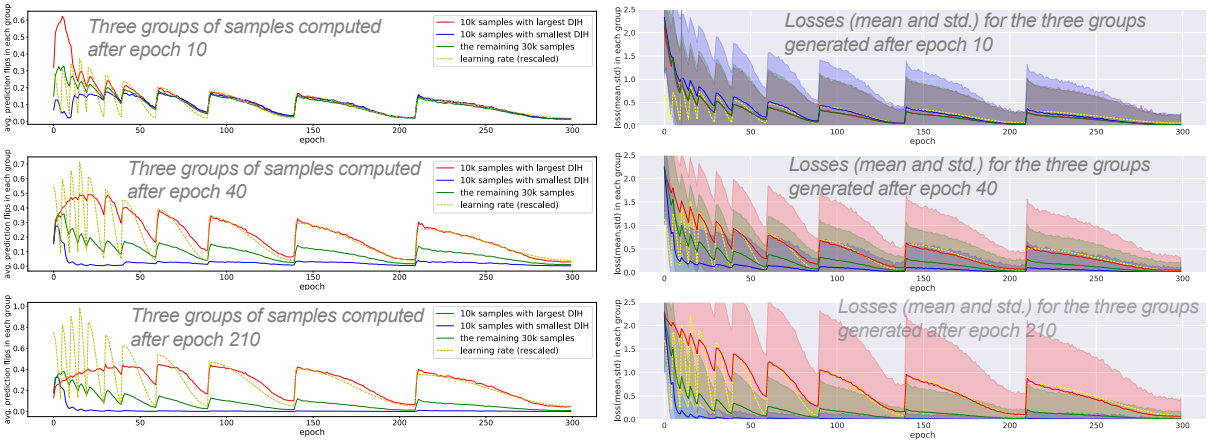

Figure 8.2: **LEFT:** Averaged prediction-flip and **RIGHT:** losses (mean and std.) of the three groups of samples partitioned by a DIH metric (i.e.,running mean of prediction-flip) computed at epoch 10 (top), 40 (middle), and 210 (bottom) during training a WideResNet-28-10 on CIFAR10. Early DIH (epoch 40) can already predict the forgettable and memorable samples at much later stages (epoch 210). The failed partition based on epoch 10 shows the importance of sufficient exploration needed for DIH to accurately measure hardness over time.

Experimental setting: When training DNNs, as shown in Figure 8.1, the instantaneous hardness measure (e.g., the per-sample loss) is usually too noisy and unstable to reflect the learning progress of the model. DIH, on the other hand, is a simple alternative descriptor of the training dynamics that averages out the noise. In the following, we use DIH as a tool to study the training dynamics of DNNs on individual samples. We train a WideResNet of depth 28 and width factor 10 on the CIFAR10 dataset by random mini-batch SGD, and apply a modified cosine annealing learning rate schedule [142] for multiple epochs of increasing length (300 epochs in total) and a decaying target learning rate. We contend that a cyclic learning rate suits our study because: (1)

it includes the most commonly used monotone decreasing schedule since the learning rate in each cycle is decreasing; (2) compared to a monotone decreasing schedule, it can uncover the dynamic properties of DIH in more scenarios such as increasing learning rates and different learning rate decay speeds. In the study, we compute DIH using two types of instantaneous instance hardness, where $a_t(i)$ is either loss or prediction flips (i.e., cases (A) or (C)). Since we do not apply any curriculum learning just yet, we always keep $S_t = V = [50000]$.

Instead of visualizing $r_t(i)$ for all $i \in [50000]$ training samples, we use $r_t(i)$ (with $a_t(i)$ being prediction flips) to categorize them into three groups, and we do this at epochs 10 (early training), 40 (middle), and 210 (later training). At epoch 40, the 10,000 samples with the largest $r_{40}(i)$ comprise the first group, the 10,000 samples ones with the smallest $r_{40}(i)$ comprise the next group, and the remaining 30,000 samples comprise the final group. We will show that the training dynamics of the three groups have different characteristics. In Figure 8.2, we plot the dynamics of the average prediction flips over each group (left plot) and the mean/standard deviation of loss in each group (right plot).

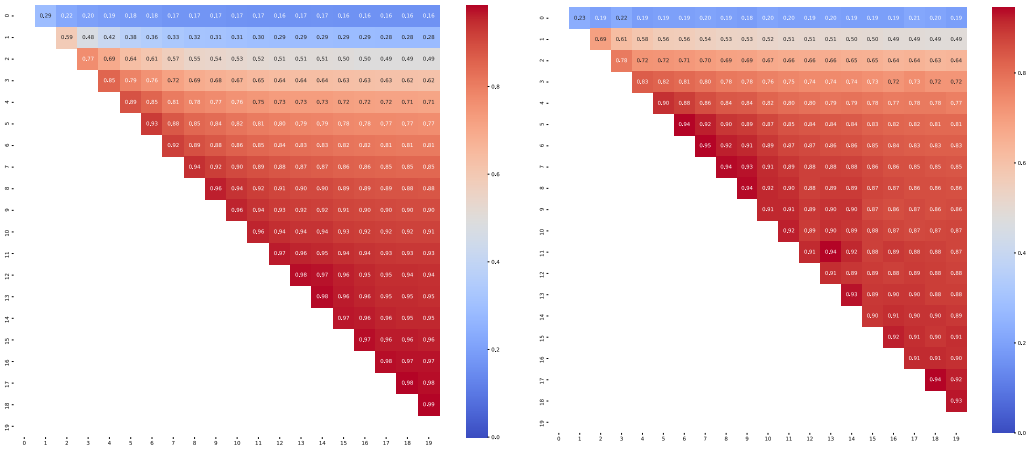

Figure 8.3: **LEFT:** Entry $A_{i,j}$ $(i < j)$ is the percentage of shared samples between the 10k samples with the **largest** DIH computed in epoch $15i$ and epoch $15j$. **RIGHT:** Entry $A_{i,j}$ $(i < j)$ is the percentage of shared samples between the 10k samples with the **smallest** DIH computed in epoch $15i$ and epoch $15j$. It shows that both the hard and easy samples in the future are predictable using the DIH values computed in early epochs.

DNNs have very different training dynamics on samples with small and large DIH. In our empirical studies, at any step-t, we observe that a group of samples with small $r_t(i)$ are quickly learned in early epochs and, thereafter, their losses remain small with predictions almost unchanged.

Since the behaviour on these samples is stationary even when the model changes by many steps with varying step sizes over the loss landscape along noisy SGD directions, the model reaches a point that is a relatively flat local minimum common amongst these samples. Moreover,it also indicates that these samples are distant from the classification boundaries and have large margins so their loss values and predictions are more stable to the changes over the course of training. This suggests that it is safe to revisit these samples less frequently. By contrast, the samples with large $r_t(i)$ show a large variance during training, i.e., their losses oscillate between small and large values and their predictions frequently change, indicating difficulty and small margins. So they tend to be the support vectors for the the local classification boundaries. Their dynamics on average trace the changes of learning rate, which implies that doing well on these samples is achieved only at relatively sharp local minima. This suggests that the model learns and generalizes better and faster on the easy samples than the hard ones.

Directly identifying these local support vectors in early stages is challenging since their nearest classification boundaries may change over the course of training by different speeds or scales. Hence, it is non-trivial to analyze the distance of each sample to its nearest local classification boundary and it is difficult to further determine the margin threshold for support vectors. DIH overcomes this problem by tracking the loss/output changes over a period of time and leveraging their statistics collected over multiple training steps to estimate the proximity of samples to the local classification boundaries. Thereby, it can extract more stationary and stable patterns hold for future steps (i.e., whether a sample tend to have a small or large margin) without requiring extra exploration on the nearby regions of each sample.

Similar to human learning [127], a natural strategy would learn the hard samples more frequently (to search for better local minima) while reducing the reencounter frequency of the easy, already learnt, ones. This can also reduce computation since it is focused more where it is needed.

We can use DIH to identify the easy and hard samples accurately, but do we need to pay the

price of training a model until convergence like [212] in order to get the DIH values? By comparing the plots with different t in Figure 8.2, we can see that **DIH in early epochs suffices to identify the easy vs. the hard samples**. The samples with small DIH at epoch 40 will remain relatively small compared to other samples even at later epochs. The hard (large DIH) samples remain hard in the future. That is, based on $r_t(i)$ (even for early stages when t is not large), we surmise that it will be prudent to apply additional training effort on hard samples and begin de-emphasizing the already learnt easy samples.

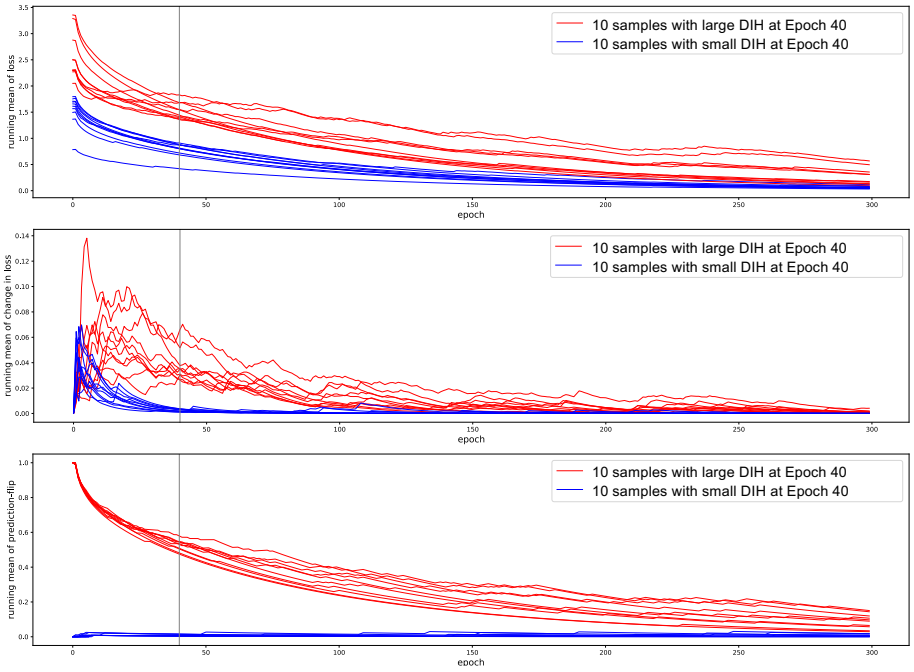

Figure 8.4: The three strategies for DIH on 10 hard and 10 easy samples, each that have been randomly sampled from the top 10k samples with the largest/smallest DIH at Epoch 40.

We empirically verify that the samples with large/small DIH in the future can be predicted by only using the DIH during early epochs. In Figure 8.3, we show the overlap rate of hard/easy samples between pairs of epochs as two upper-triangle matrices. For example, given U_i, the 10k samples with the **largest** DIH in epoch $15i$, and U_j for any $j > i$, $A_{i,j} = {}^{|U_i \cap U_j|}/_{10000}$ for the matrix

A in the left plot. Similarly, the matrix in the right plot measures the overlap rate for the 10k samples with the **smallest** DIH between epoch $15i$ and epoch $15j$. They show that after a few early epochs, DIH can accurately predict the hard and easy samples in the future. This verifies our statement in the last paragraph. In addition, it shows that $|U_i \cap U_j|/10000$ between consecutive epochs $15i$ and $15j$ is close to 100%, which suggests that DIH is a stable, consistent, and smoothly varying measure. This allows us to save computation by lazily updating DIH only on a subset of samples S_t per step during training, as we do in the definition of DIH in Eq. (8.2).

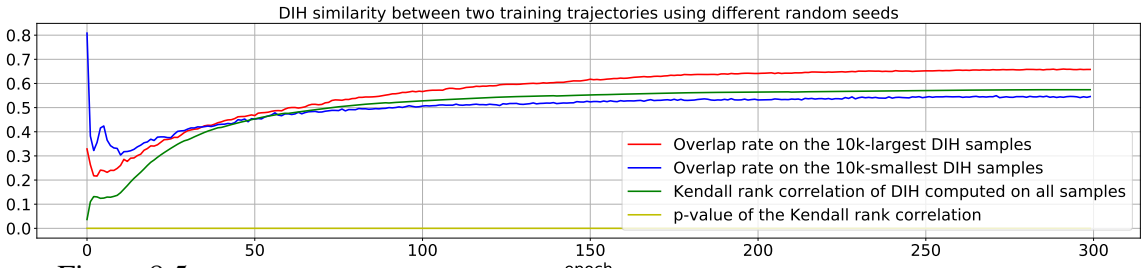

Figure 8.5: Correlation between DIH in two training experiments using different random seeds.

We find that **DIH on the same sample is robust and insensitive to the randomness of training.** To verify this, we run the above experiment twice on CIFAR10 using two different random seeds. We then compute the overlap rate between the 10k samples with the largest/smallest DIH at each epoch. Since DIH performs as a score to select/rank samples in our approach, we can use Kendall rank correlation coefficient (i.e., Kendall's τ) to evaluate the correlation between DIH values of a sample at the same epoch of the two random experiments. We report the overlap rates and Kendall's τ (with its p-value) in Figure 8.5. During training, the two overlap rates and Kendall's τ all quickly grow to > 0.6 while the p-value stays near 0. This indicates a strong correlation between the DIH of the two random trials.

Previous CL methods using instantaneous hardness need to evaluate it for all samples before selecting any sample in each step, which involves extra inference computation for the unselected samples. This is expensive when training DNNs. While switching to DIH, such extra computation

can be avoided. Figure 8.4 shows that all three types ((A), (B), and (C)) of **DIH metrics decrease during training for both easy and hard samples.** If our curriculum prefers selecting hard samples with large DIH, it is safe to update their DIH lazily, since the stale DIH values for the unselected samples are greater than their up-to-date true values, so the future steps will not miss informative samples. Updating DIH of the selected samples requires only a byproduct of back-propagation, which is already available after inference completes. This also indicates that as learning continues, samples become less informative, so that we can select and train on fewer samples.

In the following, we further report and compare the dynamics in four scenarios using plots as Figure 8.2: (1) under 100% label noise (Figure 8.6); (2) under 40% label noise (Figure 8.9); (3) training a smaller DNN (Figure 8.7); and (4) using exponential decaying learning rate across epochs (Figure 8.8).

In summary, they show that (1) the dynamics revealed by DIH is entirely different for data with incorrect labels; (2) clean and noisy labeled data exhibit different dynamics that can be distinguished by DIH; (3) DIH values across samples are more different on deeper and wider DNNs, implying that large DNNs are more data-selective in learning; and (4) DIH shows similar properties with other learning rate schedules.

In particular, **we firstly conduct an empirical study of dynamic instance hardness during training a neural net on very noisy data**, as studied in [242] and [7]. Specifically, we replace the ground truth labels of the training samples by random labels, and apply the same training setting used in Section 8.1.3. Then, we compute the running mean of prediction-flip for each sample at some epoch (i.e., 10, 40, 60), and partition the training samples into three groups, as we did to generate Figure 8.2. The result is shown in Figure 8.6. It shows (1) the group with the smallest prediction flip over history (left plot) is possible to have large but unchanging loss as shown in the right plot; and (2) the DIH in this case can only reflect the history but cannot predict the future. However, it also indicates that the capability of DIH to predict the future is potential to be an

effective metric to distinguish noisy data or adversarial attack from real data. We will discuss it in our future work.

We secondly change the WideResNet to a much smaller CNN architecture with three convolutional layers[*]. We apply the same training setting used in Section 2. Then, we compute the running mean of prediction-flip for each sample at some epoch (i.e., 10, 40, 140, 210), and partition the training samples into three groups, as we did to generate Figure 8.2. The result is shown in Figure 8.7. Compared to DIH of training deeper and wider neural nets shown in Figure 8.2, the memorable and forgettable samples are indistinguishable until very late stages, e.g., Epoch-140. This indicates that using DIH in earlier stage to select forgettable samples into curriculum might not be reliable when training small neural nets. We will leave explanation of this phenomenon to our future works.

Moreover, we provide a comparison of the smoothness between DIH and instantaneous loss on individual samples in Figure 8.1. It shows that the DIH is a smooth and consistent measure of the learning/memorization progress on individual samples. In contrast, the frequently used instantaneous loss is much noisier, so selecting training samples according to it will lead to unstable behaviors during training. In Figure 8.10, we also provide a comparison of DIH and instantaneous loss on the two groups of samples in Figure 8.4, which shows a similar phenomenon.

8.2 Scores derived from Optimizing Training Dynamics

Instead of designing a score function from the heuristics, in this section, we explore a novel methodology that derives a score function for curriculum learning from a formulation proposed in Chapter 4. In each round, the goal is to find a subset of samples such that training the model on them leads to the greatest progress and fastest learning speed towards the ground-truth on all available samples. Inspired by an analysis of optimization dynamics under gradient flow for both

[*]The "v3" network from https://github.com/jseppanen/cifar_lasagne.

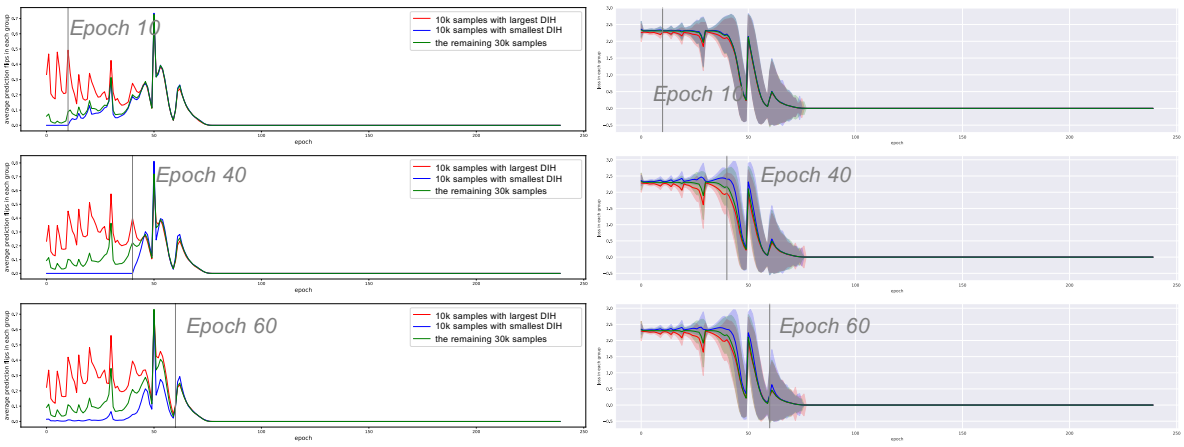

Figure 8.6: **LEFT:** Averaged prediction-flip and **RIGHT:** losses (mean and std.) of the three groups of samples partitioned by a DIH metric (i.e.,running mean of prediction-flip) computed at epoch 10,40 and 60 during training WideResNet-28-10 on CIFAR10 with **random labels**. In this setting, the random (but wrong) labels will be remembered very well after some training, and DIH in early stages loses the capability to predict the future DIH, i.e., they can only reflect the history but not the future. This characteristic of DIH might be helpful to detect noisy data.

regression and classification in Chapter 4, the problem reduces to selecting training samples by a score computed from samples' residual and linear temporal dynamics. It encourages the model to focus on the samples at the learning frontier, i.e., those with large loss but fast learning speed. The scores in discrete time can be estimated via already-available byproducts of training, and thus require negligible extra compute.

Our score matches the intuition to always select data at the learning frontier, i.e, hard samples that the model is making the greatest progress on, while previous work mainly focus on the hard samples even some of them are outliers inconsistent with other samples. Although the DIH score discussed in Section 8.1 can partially capture either of them by choosing different instantaneous feedback (i.e., loss or loss change), it does not take both into account in a score function.

In particular, the first quantity of our score is the sample's prediction residual, and encourages the selection of samples that predict far away from their targets. Similar criteria have been studied in active learning, boosting, and curriculum learning [248, 98]. By focusing on data with large residuals, we do not waste computation on samples that are already learnt. The second quantity is

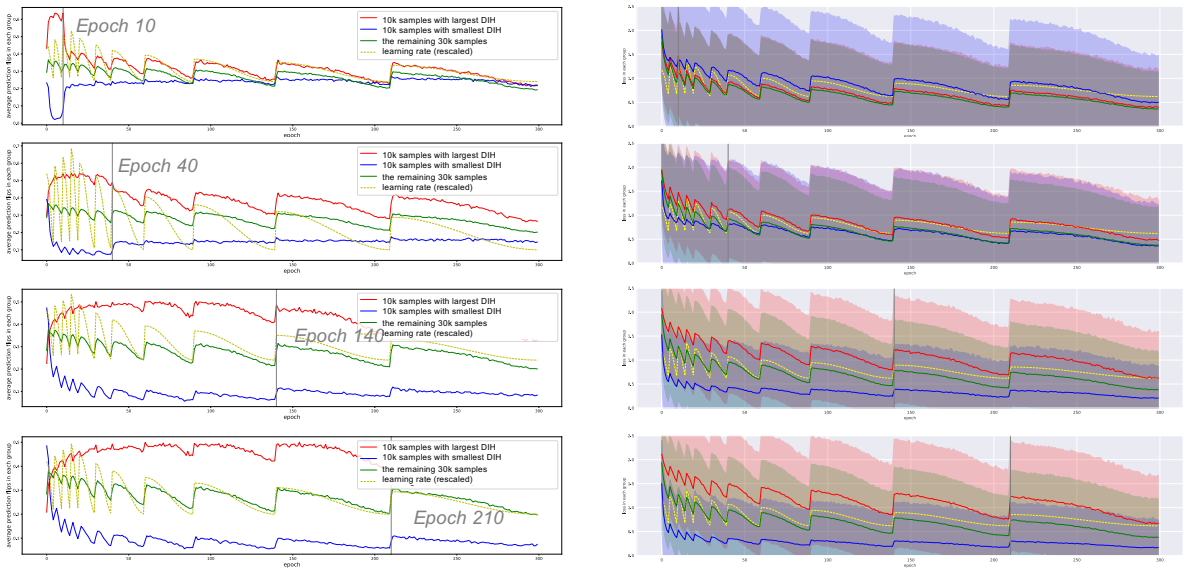

Figure 8.7: **LEFT:** Averaged prediction-flip and **RIGHT:** losses (mean and std.) of the three groups of samples partitioned by a DIH metric (i.e.,running mean of prediction-flip) computed at epoch 10, 40, 140 and 210 during training **a smaller CNN** on CIFAR10. It shows that the difference of memorable and forgettable samples is not sufficiently obvious until very late training epochs, e.g., after epoch-140.

the sample's linear dynamics under the gradient flow over the data distribution, i.e., the learning speed. Empirical deep neural network (DNN) studies [212, 253, 256] show that predictions for some samples remain fixed and correct (i.e., memorized) once learnt, while some samples' predictions frequently change during training and are easier to be forgotten. Moreover, they show that training on the latter minimally impacts the former's predictions, so we can focus training only on the latter for better efficiency. Our score is mathematically derived to be a combination of the two selection criteria formerly motivated by empirical studies and heuristics, and hence bridges the gap between theoretical principles of curriculum design and empirical observations of training dynamics.

We further discuss a natural interpretation of our score achieved when relating it to the neural tangent kernel (NTK) [95, 5, 54]. Its properties in the NTK regime also suggest the feasibility of a lazy update and moving average of the scores. We show that linear dynamics capture the gradient similarity between samples. Intuitively, applying gradient descent on samples with large linear

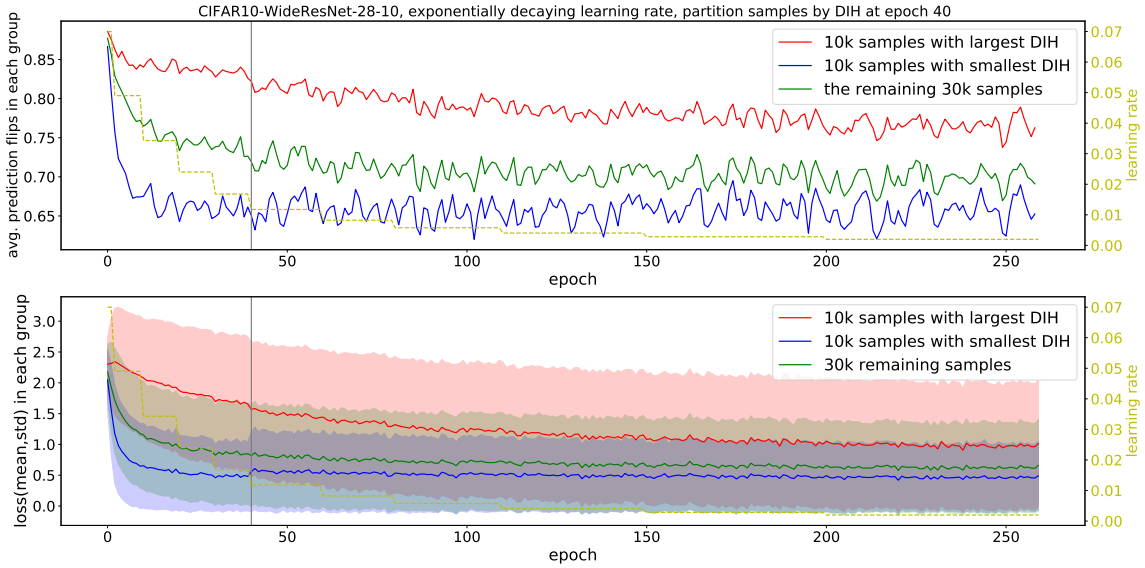

Figure 8.8: **TOP:** Averaged prediction-flip and **RIGHT:** losses (mean and std.) of the three groups of samples partitioned by a DIH metric (i.e., running mean of prediction flip) computed at epoch 40 when using **exponential decaying learning rate** (instead of cyclic cosine annealing rate) across epochs (cycles). DIH exhibits similar properties on identifying hard and easy samples for neural nets to learn.

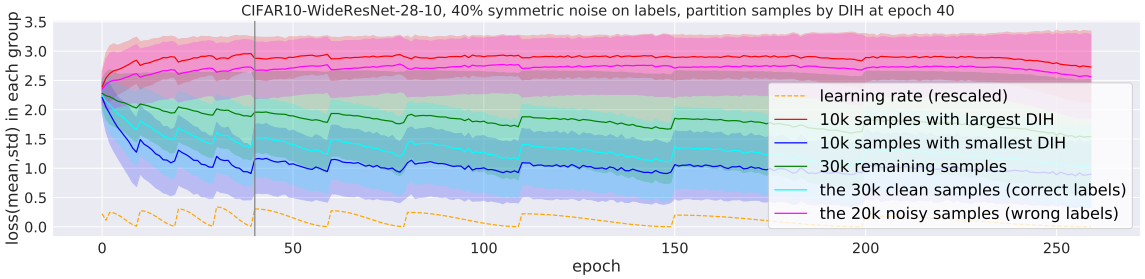

Figure 8.9: Losses (mean and std.) of the three groups of samples partitioned by a DIH metric (i.e., running mean of prediction flip) computed at epoch 40 when 40% of labels are randomly changed to another wrong class (i.e., **40% symmetric noises on labels**. We also show the losses on the clean samples with correct labels and noisy samples with wrong labels, where the former exhibit lower DIH than the latter. Hence, DIH is robust to label noises and can identify the hard and easy samples, which are mainly composed of the clean and noisy data respectively in this scenario.

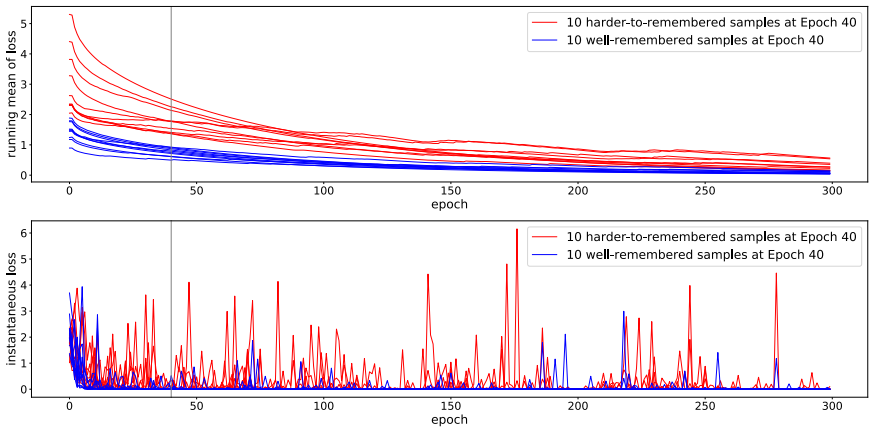

Figure 8.10: **Top:** DIH (running mean of loss) vs. **Bottom:** instantaneous loss of 10 samples randomly selected from the top 10k samples with the largest(red) and the smallest(blue) DIH at epoch 40 of training of WideResNet-28-10 on CIFAR10 (the same as Figure 8.4. It shows that for each individual sample from the two groups, DIH smoothly decreases while the corresponding instantaneous loss is much noisier.

dynamics can effectively reduce losses on many similar samples and may further stabilize their dynamics to reach flat minima. By selecting samples with higher scores, we focus on unlearnt data whose gradients are most consistent with gradients of other data. Hence, the selected samples have significant impacts on the optimization process and by focusing on them we potentially shorten the optimization trajectory.

8.2.1 Related Work

Most score functions in previous work of curriculum learning are either given by human experts [19, 106, 12, 197] or defined by heuristic criteria [207, 201, 208, 71, 74], e.g., hardness [116] or representativeness [100, 248] of samples. However, these criteria might not necessarily be directly related to the original training objective. Some of them suffer from hyperparameter sensitivity, e.g., a threshold on loss values. Although the ultimate goal of CL (i.e., finding an optimal sequence of training samples) is more general than other data selection methods, CL strategies are usually built upon relatively simple heuristics without having a complete mathematical analysis.

In addition, the selection criteria of these methods were developed for various learning settings and hypothesis class assumptions, and thus can sometimes be contradictory. For example, active

learning and boosting both favor difficult-to-learn samples, while many CL methods prefer easy-to-learn samples [101, 116]. Although selection criteria are often partially adaptive to per-sample feedback during training, they are not designed to directly accelerate the learning process, as we do in this section. Some recent work [102, 59] resorts to an additional model to directly generate selection results but they require training another model using non-stationary feedback from the ongoing training process, e.g., via reinforcement learning, which might be more challenging and costly to solve than the original problem.

A line of recent research [199, 233] has studied accelerated optimization dynamics derived from discretized Lagrangian/Hamiltonian dynamics of a model, showing optimal convergence rates. By doing so, they recover a class of accelerated optimization schemes and even generate new ones. These approaches mainly focus on convex optimization. The major difference with our work is that we optimize the dynamics of a model's output on individual samples (vs. on model parameters) by changing the training set (vs. by choosing kinetic energy function, scaling conditions and discretization). In addition, they optimize the total energy along the optimization trajectory, which might be an objective worth studying for our problem in the future.

8.2.2 Regression

To optimize Eq. (4.8) proposed in Chapter 4,, we approximate the expected momentum w.r.t. $x \sim \mathcal{D}$ in Eq. (4.8) by averaging over a finite number of samples D drawn from the data distribution $\mathcal{D}$. This leads to the following approximated objective derived in Eq. (4.10), i.e.,

$$\mathbb{E}_{x\sim\mathcal{D}}\left\langle y - f(x), \left.\frac{\partial f(x)}{\partial t}\right|_S \right\rangle \approx \frac{1}{|D|}\sum_{i\in S}\left\langle y_i - f(x_i), \left.\frac{\partial f(x_i)}{\partial t}\right|_D \right\rangle. \tag{8.3}$$

This introduces a per-sample score $a_t(i)$ as the inner product of two vectors at step t, i.e., the residual $y_i - f(x_i)$ and its dynamics under the gradient flow computed on D:

$$a_t(i) \triangleq \left\langle y_i - f(x_i;\theta_t), \left.\frac{\partial f(x_i;\theta_t)}{\partial t}\right|_D \right\rangle. \tag{8.4}$$

Hence, the expectation in Eq. (4.8) can be approximated by a function that sums up the scores of all selected samples $i \in S$. Note the two vectors can be directly obtained from the byproduct of training on S and D, so estimating the score for all the candidate samples does not require any additional computation. However, in each step of curriculum learning, we only train the model on a subset S and only update the score for $i \in S$. In practice, this problem can be mitigated by maintaining an exponential moving average $\hat{a}_{t+1}(i)$ of $a_t(i)$ over time:

$$\hat{a}_{t+1}(i) = \begin{cases} \gamma \times \hat{a}_t(i) + (1-\gamma) \times a_t(i) & \text{if } i \in S_t \\ \hat{a}_t(i) & \text{otherwise,} \end{cases} \tag{8.5}$$

As we will discuss later, with sufficient exploration over all samples and in the regime of DNN training, $\hat{a}_{t+1}(i)$ is a high-quality and more stable alternative to $a_t(i)$ that is almost free to compute. According to Eq. (4.8), the optimal S_t simply selects the top-k samples with the largest scores. However, S_t cannot replace D in estimating the linear dynamics $\left.{\partial f(x_i)}/{\partial t}\right|_D$ because S_t can be biased relative to the data distribution $\mathcal{D}$. Therefore, in our algorithm presented later, instead of selecting the top-k, we sample S_t based on their scores, and cyclically employ a large-batch training epoch over uniform samples from the training set after every episode of mini-batch training on the selected subsets. These strategies encourage more exploration for better estimate to the scores in practice.

Remarks: Take a closer look at the induced score at the end of Eq. (4.10): the residual $y_i - f(x_i)$ measures the gap between the current prediction $f(x_i)$ and the ground truth y_i (i.e., how hard the sample is), while the linear dynamics delineates how $f(x_i)$ changes (i.e., speed and direction) when

training the model using samples drawn from the data distribution. Together, their inner product reflects the momentum of $f(x_i)$ moving towards y_i under the gradient flow on D. Intuitively, we tend to select (1) harder samples that the model can make more progress on and (2) samples that are consistent with most other samples drawn from the same distribution (indicating that reducing the losses on D helps to also move $f(x_i)$ towards y_i). The former intends to select the most informative ones (compared to the ones already learned) and is consistent with the selection criteria proved to be effective in previous curriculum learning [248, 253] and boosting methods [185, 65], while the latter tends to select the most representative ones that are consistent with other data, which is another criterion whose success has been demonstrated in recent curriculum learning methods [254, 256]. However, unlike many previous criteria that are built upon empirical observations or human heuristics, Eq. (4.10) is derived from a well-formulated and motivated optimization problem.

8.2.3 Classification

We can extend the above score function for regression to the general multi-class classification task. By approximating the expectation with samples D drawn from $\mathcal{D}$, we have the result presented in Eq. (4.13), i.e.,

$$\mathbb{E}_{x\sim\mathcal{D}}\left\langle \mathbf{y}-p(x),\left.\frac{\partial p(x)}{\partial t}\right|_S\right\rangle \approx \frac{1}{|D|}\sum_{i\in S}\left\langle \mathbf{y}_i-p(x_i),\left.\frac{\partial f(x_i)}{\partial t}\right|_D\right\rangle. \tag{8.6}$$

Hence, we compute the per-sample score $a_t(i)$ by

$$a_t(i)\triangleq\left\langle \mathbf{y}_i-p(x_i;\theta_t),\left.\frac{\partial f(x_i;\theta_t)}{\partial t}\right|_D\right\rangle, \tag{8.7}$$

Which has a form similar to Eq. (8.4) except that the residual is $\mathbf{y}_i-p(x_i;\theta_t)$ for classification. The linear dynamics term in Eq. (8.7) is associated with the gradient flow minimizing the L2 loss $\ell(\cdot)$ on D instead of the cross-entropy loss $\ell_{xe}(\cdot)$ on S. This is the major difference between Eq. (8.7) and

Eq. (8.4), which uses the same loss $\ell(\cdot)$ for both the model training and dynamics estimation. This difference requires the training steps to switch between the two types of losses, i.e., we minimize the cross-entropy loss $\ell_{xe}(\cdot)$ during mini-batch training on S_t and switch to the square loss $\ell(\cdot)$ in the large-batch training epoch on $D \sim \mathcal{D}$ at the end of each episode/cycle.

8.2.4 Connections to Neural Tangent Kernel

We can obtain an intuitive explanation of the score in Eq. (4.10) under the context of neural tangent kernel (NTK) [95, 54]. For simplicity, we focus on the regression task in the single-output case (the result can be extended to every dimension in the multiple-output case). The second row of Eq. (4.10) can be written as

$$
\begin{aligned}
&\mathbb{E}_{x\sim\mathcal{D}}\left\langle y-f(x), \left.\frac{\partial f(x)}{\partial t}\right|_S \right\rangle \\
\approx&\frac{1}{|D|}\sum_{i\in S} -\frac{\partial \ell(x_i)}{\partial f(x_i)} \cdot \sum_{x\in D}\left\langle \frac{\partial f(x_i)}{\partial \theta}, \frac{\partial f(x)}{\partial \theta}\right\rangle \cdot [y-f(x)] \\
=&\frac{1}{|D|}\sum_{i\in S} \frac{\partial \ell(x_i)}{\partial f(x_i)} \cdot \sum_{x\in D}\left\langle \frac{\partial f(x_i)}{\partial \theta}, \frac{\partial f(x)}{\partial \theta}\right\rangle \cdot \frac{\partial \ell(x)}{\partial f(x)} \\
=&\frac{1}{|D|} r_S^T H_{S,D} r_D = \frac{1}{|D|}\sum_{i\in S, j\in D} H_{i,j} r_i r_j, \\
&r_i \triangleq \frac{\partial \ell(x_i)}{\partial f(x_i)} = f(x_i) - y_i, \;\; H_{i,j} \triangleq \left\langle \frac{\partial f(x_i)}{\partial \theta}, \frac{\partial f(x_j)}{\partial \theta}\right\rangle.
\end{aligned}
\tag{8.8}
$$

One can think that H is a dynamic kernel matrix describing the pairwise relationship between sample-i and sample-j in terms of their model gradients at step t. Note both r and H depend on θ_t so they are time-variant. In recent work [95, 5], it is shown that when $f(\cdot)$ is a neural network with enough neurons per layer (i.e., with adequate but still finite width), with high probability, H converges to a deterministic kernel matrix H^* so-called the "neural tangent kernel (NTK)" computed on random initialization. In this case, our objective becomes a weighted sum of the pairwise product of residuals

$r_i r_j$ over all $i \in S, j \in D$, where the weights are time-invariant and determined by H^*, i.e.,

$$\mathbb{E}_{x\sim\mathcal{D}}\left\langle y - f(x), \frac{\partial f(x)}{\partial t}\Big|_S \right\rangle \to \frac{1}{|D|}\sum_{i\in S, j\in D} H^*_{i,j} r_i r_j = \frac{1}{|D|}\sum_{i\in S}\left[\sum_{j\in D} H^*_{i,j} r_j\right]\cdot r_i. \tag{8.9}$$

Given the NTK H^*, which is a static matrix describing the pairwise correlation between samples, we can obtain more insights about dynamics optimization in Eq. (4.8). First, setting S to be all the training samples, i.e., $S = [n]$, is not guaranteed to maximize the objective in Eq. (8.9). Instead, it prefers samples with both large (i.e., large in magnitude) residuals r_i and strong correlations to other samples with large residual r_j. Specifically, the objective tends to select difficult samples (i.e., large $|r_i|$) that are representative of (i.e., $\operatorname{sign}(H_{i,j}) = \operatorname{sign}(r_i r_j)$) and strongly related to (i.e., large $|H_{i,j}|$) other difficult samples $j \in D$ (i.e., large $|r_j|$). Such criteria rule out the following two types of samples, which might be selected by previous curricula: (1) difficult samples with large residuals but weakly related to other samples, which can possibly be outliers (or adversarially chosen) that fail on training; (2) easy samples with small residuals that can only contribute very weak gradients to improve the predictions on difficult samples.

Furthermore, in the NTK regime, H^* does not change over time, so the score of each sample x_i solely depends on its own residual r_i and the residual r_j of its strongly related samples from D. Hence, when applied to training over-parameterized (wide-enough) neural nets, the objective tends to keep selecting the same x_i until most of the strongly-related-samples to x_i have sufficiently small residuals or r_i itself becomes nearly zero. If samples can be well structured by H^*, e.g., H^* has a block diagonal structure after certain symmetric row/column permutation where each block forms a cluster, the dynamics optimization will keep reducing the errors on some clusters until their errors become sufficiently small before switching to other clusters. This property allows us, in practice, to lazily update the scores (which requires large-batch training on i.i.d. samples $D \sim \mathcal{D}$ and might degenerate performance), and for most other steps we can still train the model via mini-batch SGD

on the selected subset S_t. That being said, a static H^* is not required by DoCL: the lazy update should work well if the block diagonal structure of H does not change too quickly. In addition, **the score computation in DoCL does not require explicitly computing** H^*. In fact, we avoid additional heavy computation by using only already-computed byproducts of the training process to estimate the linear dynamics in Eq. (4.10).

8.3 Time Consistency of Unlabeled Data

In this section, we study the dynamics of neural net outputs during semi-supervised learning (SSL). We analyze possible catastrophic forgetting on labeled data caused by adding an unlabeled sample to training, from which we derive a "time-consistency (TC)" score $c^t(x)$ for an individual sample x at training step[†] t that can be used to select informative unlabeled data with more time-consistent pseudo targets in self-supervision. Specifically, $c^t(x)$ is an negative exponential moving average of $a^t(x)$ (defined below) over training history before t:

$$a^t(x) \triangleq D_{KL}(p^{t-1}(x)||p^t(x)) + \left|\log \frac{p^{t-1}_{y^{t-1}(x)}(x)}{p^t_{y^{t-1}(x)}(x)}\right|, \tag{8.10}$$

where $D_{KL}(\cdot||\cdot)$ is the Kullback–Leibler divergence, $p^t(x)$ and $y^t(x) \triangleq \operatorname{argmax}_y p^t_y(x)$ are the output distribution over classes and the predicted class label of x at step t. Intuitively, the KL-divergence between output distributions measures how consistent the output is between two consecutive steps, while the log odds ratio for the predicted class $y^{t-1}(x)$ measures the change of confidence on the predicted class $y^{t-1}(x)$. A moving average of $a^t(x)$ naturally captures inconsistency over time quantify, based on the history, how much change will occur to the learning objective when selecting x and its pseudo target for future training.

We define the time-consistency (TC) of a sample x at step t as an exponential moving average

[†]We use superscripts to index the training step.

of $-a^t(x)$ over time, i.e.,

$$c^t(x) = \gamma_c(-a^t(x)) + (1-\gamma_c)c^{t-1}(x) \tag{8.11}$$

where $\gamma_c \in [0,1]$ is a discount factor. We negate $a^t(x)$ so that larger $c^t(x)$ means better time-consistency.

8.3.1 Theoretical Justification

For classification tasks, we show that the changes in labeled samples' losses are bounded by $a^t(x)$ on unlabeled data. In other words, choosing time-consistent unlabeled samples mitigates catastrophic forgetting of labeled sample.

We denote the labeled data set as $\mathscr{L}$, an unlabeled data set as $\mathscr{U}$, and an unlabeled sample as $x' \in \mathscr{U}$. Let $f^t(x)$ be the final layer output (before softmax) on sample x with network parameters θ^t at step t of training, so $p^t(x) = \text{softmax}(f^t(x))$. Let $\boldsymbol{y}(x)$ be the one-hot label vector for sample x, and $\ell(x;\theta^t)$ be the cross entropy loss between the class label and the softmax output of network with parameter θ^t. We consider two cases: (1) we train the network using only the labeled samples as a gradient step, i.e., $\theta^{t+1} = \theta^t + \eta \sum_{x\in\mathscr{L}} \nabla_\theta \ell(x;\theta^t)$, where η is the learning rate; and (2) we add an unlabeled sample x' to the gradient step, i.e., $\hat{\theta}^{t+1} = \theta^t + \eta(\sum_{x\in\mathscr{L}} \nabla_\theta \ell(x;\theta^t) + \nabla_\theta \ell(x';\theta^t))$. In (2), when calculating $\ell(x';\theta^t)$, we use a one-hot label $\boldsymbol{y}^t(x')$ that has value one in position $y^t(x')$ (recall $y^t(x') = \text{argmax}_j\, p_j^t(x')$, where $p_j^t(x')$ is class-j's probability in distribution $p^t(x')$) and value zero elsewhere — which is a "winner take all" or a "pseudo" target. The Taylor expansion of labeled sample loss $\ell(x,\theta)$ defined on θ_a and evaluated at θ_b is:

$$g_{\theta_a}(\theta_b) = \Big[\sum\nolimits_{x\in\mathscr{L}} \ell(x;\theta_a) + \nabla_\theta \ell(x;\theta_a)(\theta_b - \theta_a)\Big] + o((\theta_b-\theta_a)^2) \tag{8.12}$$

We can measure the forgetting effect of adding $x' \in \mathscr{U}$ to the training set by looking at the changes

in loss over labeled samples $x \in \mathscr{L}$. Ideally, x' should not cause vital changes to the loss over labeled samples, which should remain small. If θ_a is close to θ_b, it is reasonable to omit the second and higher order terms of the Taylor expansion in Eq. (8.12). Doing so, we calculate the change of loss for labeled data by

$$\begin{aligned}\frac{1}{\eta}\left|\sum_{x\in\mathscr{L}}\left[\ell(x;\theta^{t+1})-\ell(x;\hat{\theta}^{t+1})\right]\right| =& \frac{1}{\eta}\left|g_{\theta^t}(\theta^{t+1})-g_{\theta^t}(\hat{\theta}^{t+1})\right| \approx \left|\nabla_\theta \ell(x';\theta^t)\sum_{x\in\mathscr{L}}\nabla_\theta \ell(x;\theta^t)\right| \\ \approx& \left|\frac{\partial \ell(x';\theta^t)}{\partial \theta^t}\frac{\partial \theta^t}{\partial t}\right| = \left|\frac{\partial \ell(x';\theta^t)}{\partial t}\right| = \left|\frac{\partial \ell(x';\theta^t)}{\partial f^t(x')}\frac{\partial f^t(x')}{\partial t}\right| \\ =& \left|(\boldsymbol{y}^t(x')-p^t(x'))\frac{\partial f^t(x')}{\partial t}\right| \end{aligned} \tag{8.13}$$

If the learning rate is not too large, we may use the difference $(f^{t+1}(x')-f^t(x'))$ to approximate

$\partial f^t(x')/\partial t$. After some algebra detailed below, let $z = y^t(x')$, we have:

$$\left|(\boldsymbol{y}^t(x') - p^t(x'))\frac{\partial f^t(x')}{\partial t}\right| \tag{8.14}$$

$$\approx \left|(\boldsymbol{y}^t(x') - p^t(x'))(f^{t+1}(x') - f^t(x'))\right| \tag{8.15}$$

$$= \left|(1 - p_z^t(x'))(f_z^{t+1}(x') - f_z^t(x')) - \sum_{j\neq z} p_j^t(x')(f_j^{t+1}(x') - f_j^t(x'))\right| \tag{8.16}$$

$$= \left|-\left[\sum_j p_j^t(f_j^{t+1}(x') - f_j^t(x'))\right] + (f_z^{t+1}(x') - f_z^t(x'))\right| \tag{8.17}$$

$$\begin{aligned} = \Bigg|\Bigg[&-\sum_j p_j^t(x') \log \frac{\exp(f_j^{t+1}(x'))}{\exp(f_j^t(x'))} - p_j^t(x') \log \frac{\sum_i \exp(f_i^t(x'))}{\sum_i \exp(f_i^{t+1}(x'))}\Bigg] \\ &+ \log \frac{\sum_i \exp(f_i^t(x'))}{\sum_i \exp(f_i^{t+1}(x'))} \sum_j p_j^t(x') + (f_z^{t+1}(x') - f_z^t(x'))\Bigg| \end{aligned} \tag{8.18}$$

$$= \left|D_{KL}(p^t(x')||p^{t+1}(x')) + \log \frac{\sum_i \exp(f_i^t(x'))}{\sum_i \exp(f_i^{t+1}(x'))} + (f_z^{t+1}(x') - f_z^t(x'))\right| \tag{8.19}$$

$$\begin{aligned} = \Bigg|&D_{KL}(p^t(x')||p^{t+1}(x')) + (\log \sum_i \exp(f_i^t(x')) - \log \exp(f_z^t(x'))) \\ &+(\log \exp(f_z^{t+1}(x')) - \sum_i \exp(f_i^{t+1}(x')))\Bigg| \end{aligned} \tag{8.20}$$

$$= \left|D_{KL}(p^t(x')||p^{t+1}(x')) + \log \frac{p_z^{t+1}(x')}{p_z^t(x')}\right| \tag{8.21}$$

$$\leq D_{KL}(p^t(x')||p^{t+1}(x')) + \left|\log \frac{p_z^{t+1}(x')}{p_z^t(x')}\right|. \tag{8.22}$$

we can bound the changes in loss for labeled samples by $a^t(x')$:

$$\begin{aligned} \left|(\boldsymbol{y}^t(x') - p^t(x'))\frac{\partial f^t(x')}{\partial t}\right| &\approx \left|(\boldsymbol{y}^t(x') - p^t(x'))(f^{t+1}(x') - f^t(x'))\right| \\ &\leq D_{KL}(p^t(x')||p^{t+1}(x')) + \left|\log \frac{p_{y^t(x')}^{t+1}(x')}{p_{y^t(x')}^t(x')}\right| = a^t(x'). \end{aligned} \tag{8.23}$$

Therefore, by selecting an unlabeled sample x' with high TC $c^t(x')$ (a smoothed estimate of $-a^t(x')$

using exponential moving average to eliminates noise), we will not suffer from a rapid surge on the labeled sample loss if $\sum_{x \in \mathcal{L}} \ell(x; \theta^t) \approx 0$ at step t. Thereby, x' and its pseudo target will not result in catastrophic forgetting of learned samples. On the other hand, for x' itself, we expect its pseudo target and objective to be consistent after being added to training. A large $c^t(x')$ indicates that both $p^t(x')$ and the confidence on pseudo-class $y^t(x')$ vary little over training history. Note the above analysis of forgetting effect on labeled data also extends and can be recursively applied to the unlabeled data that have been correctly predicted and selected for training. Hence, selecting unlabeled samples with large TC can avoid catastrophic forgetting of learned data.

8.3.2 Empirical Evidence

We empirically investigate the quality of unlabeled samples selected by the time-consistency metric. Ideally, in SSL training, we expect the selected samples to have pseudo targets no different from the unknown ground truths. On CIFAR-10, we randomly select 15000 samples for training and use the remaining 35000 samples for validation, where the latter is considered as unlabeled data in the SSL setting. We train a WideResNet-28-2 [240] (28 layers, width factor of 2, 1.5-million parameters) on the 15000 samples by fully supervised training, and inspect the 35000 samples to check if the time-consistency metric can effectively prune out the samples with wrong predictions on the model while selecting samples with the right labels. We compare time-consistency against confidence as the metric to select unlabeled samples in Figures 8.11,8.12 and 8.13. More empirical studies with different parameters and on CIFAR-100 can be found in Section 8.3.3, which show patterns similar to what we present here.

We report the number of correct/incorrect predictions over the validation samples selected by various thresholds on TC in Figure 8.11 and compare it with confidence $\max_y p(y|x)$, i.e., the highest probability among classes. We normalize TC and confidence values to be in $[0, 100]$ and so, the threshold set to 20 means that we select the validation samples whose TC is greater than

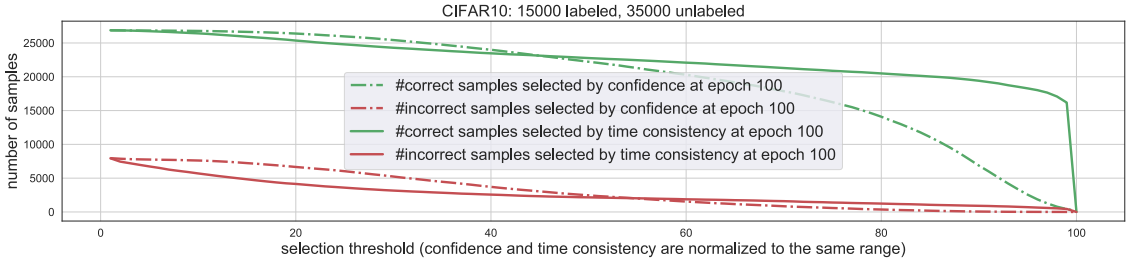

Figure 8.11: Computed time-consistency and confidence at epoch 100. The x-axis shows the validation samples selected using different thresholds on the two metrics (normalized to $[0, 100]$). The y-axis reports correct v.s. incorrect predictions over the selected samples.

20% of the maximum TC value. At high thresholds, time-consistency can more effectively identify the correctly predicted samples than confidence.

In Figure 8.12, we select the top 1000 and bottom 1000 samples in the validation set based on the two criteria computed at epoch 100 and report the performance on those samples across training epochs. Compared to confidence, the performance on the top 1000 samples selected by TC is significantly better than the bottom 1000 ones across most epochs. This again suggests that TC is a more powerful criteria for identifying correctly-predicted samples. It also shows that TC can be predictive for future training dynamics.

In Figure 8.13, we plot all incorrectly predicted samples in the validation set based on their performance and selection criteria (either TC or confidence) computed at epoch 100. Comparing the two, we find many fewer samples with high TC that have low true-class probability than we find using the confidence (see the lower-right regions in Figure 8.13). This suggests that TC could potentially prune incorrectly-predicted samples more effectively.

8.3.3 Additional Empirical Studies

We conduct the same empirical studies on CIFAR-10 dataset with the selection metrics computed at various epochs. We show epoch 50 results in Figures 8.14, 8.15 and 8.16 and epoch 200 results in Figures 8.17, 8.18 and 8.19. We observe similar patterns as discussed in Section 8.3.2

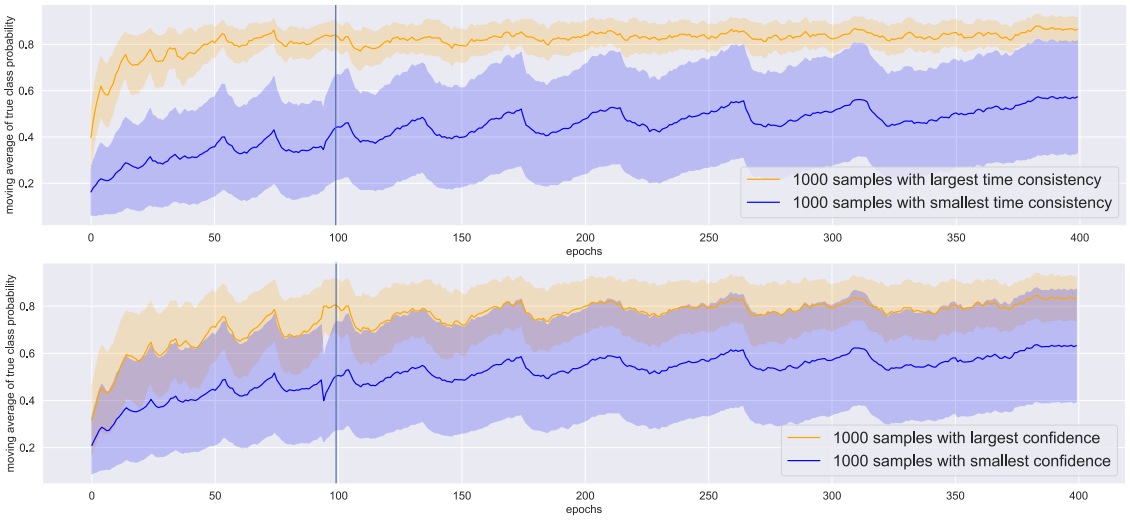

Figure 8.12: Computed time-consistency (top) and confidence (bottom) at epoch 100. Select the top 1000 and bottom 1000 validation samples based on the two metrics. Compare the moving average of true class probability of the selected samples across epochs.

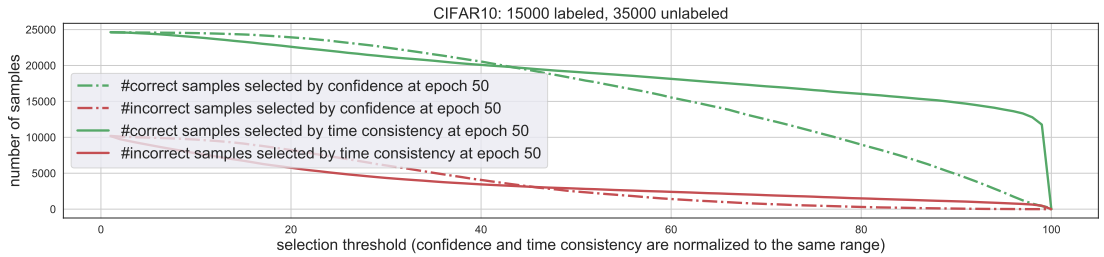

Figure 8.14: Compute time-consistency and confidence at epoch 50. Select samples in the validation set based on different thresholds of the two metrics and report the number of correct v.s. incorrect predictions in the selected samples. The two metrics are normalized to $[0, 1]$ range.

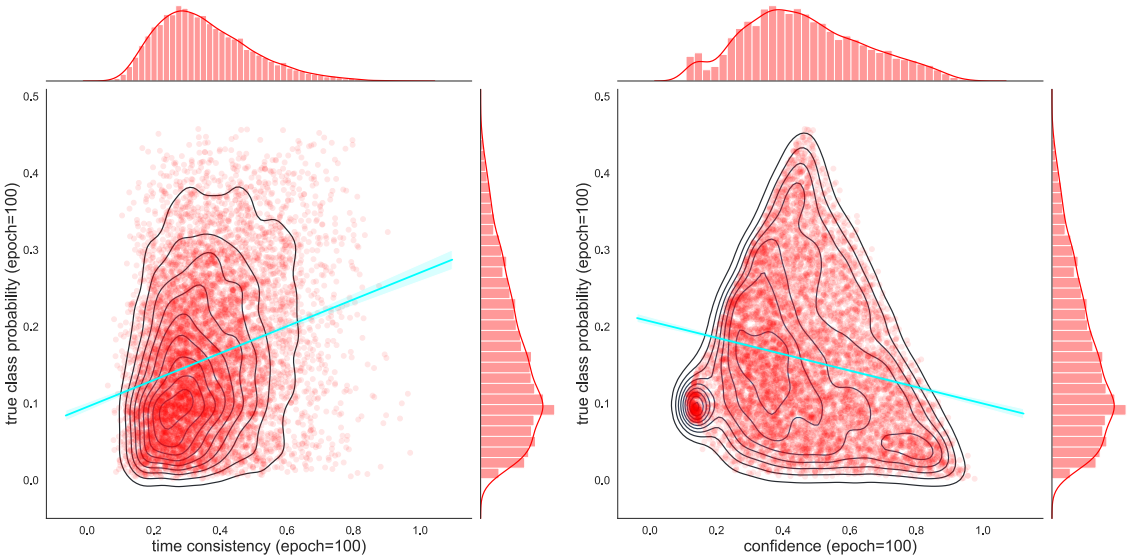

Figure 8.13: Scatter plot and correlation (cyan lines) of the true class's probability (y-axis) for incorrectly predicted validation samples with TC (**left**) or confidence (**right**) on the x-axis at epoch 100.

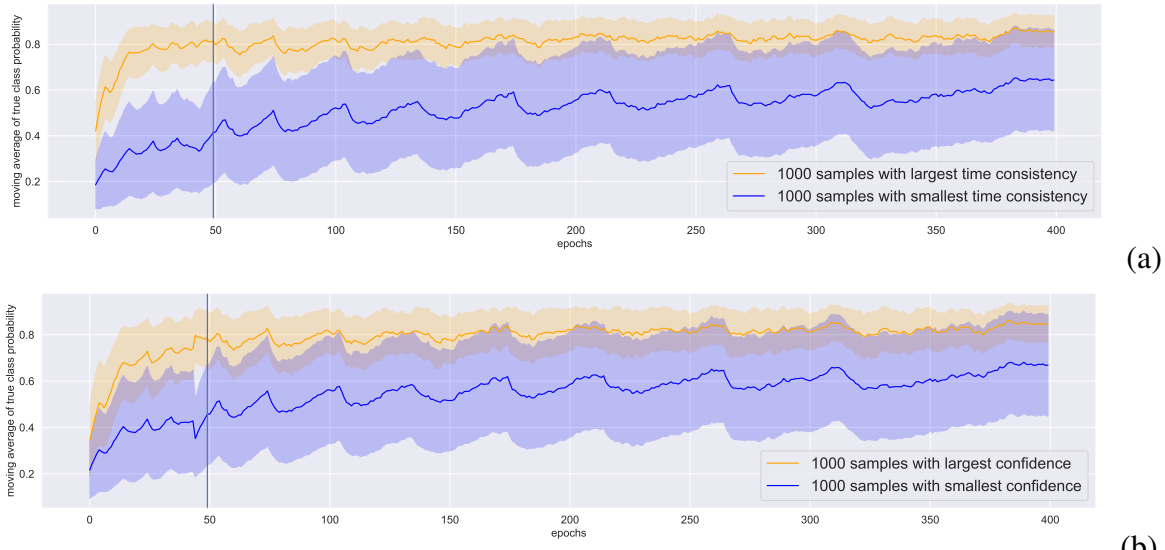

Figure 8.15: Compute time-consistency (a) and confidence (b) at epoch 50. Select the top 1000 and bottom 1000 samples in the validation set based on the two metrics. Compare the moving average of true class probability of the selected samples across training epochs.

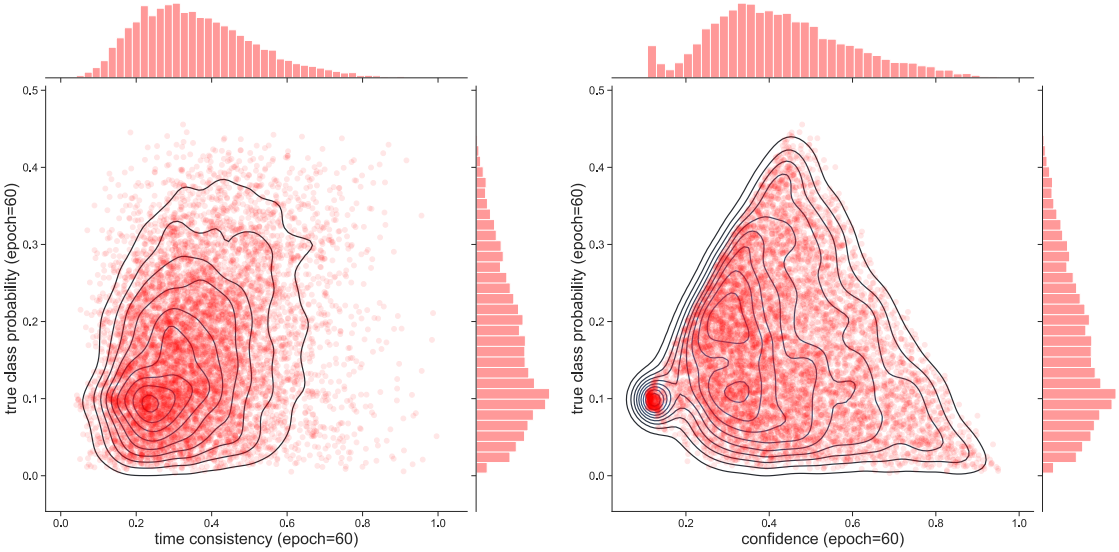

Figure 8.16: Compute time-consistency and confidence at epoch 60. Plot the incorrectly predicted samples in the validation set as a scatter plot with the selection metric as the x-axis and the true class probability as the y-axis.

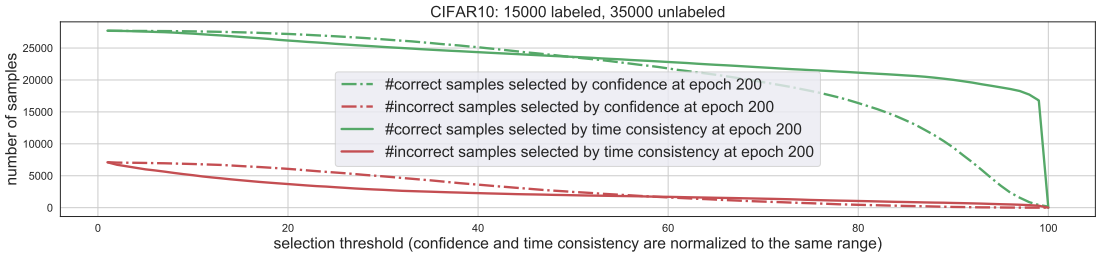

Figure 8.17: Compute time-consistency and confidence at epoch 200. Select samples in the validation set based on different thresholds of the two metrics and report the number of correct v.s. incorrect predictions in the selected samples. The two metrics are normalized to $[0, 1]$ range.

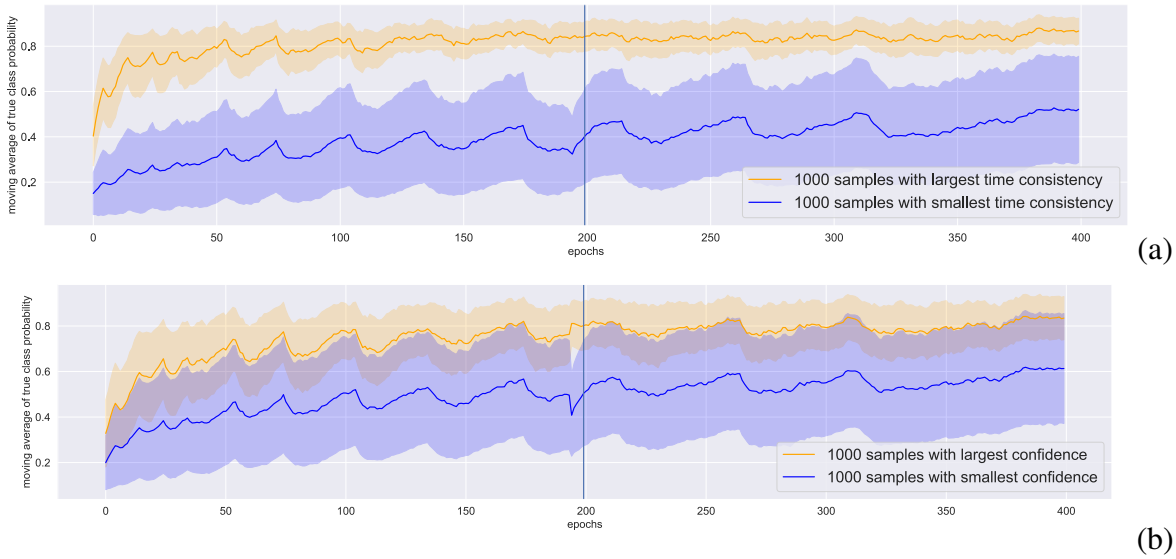

Figure 8.18: Compute time-consistency (a) and confidence (b) at epoch 200. Select the top 1000 and bottom 1000 samples in the validation set based on the two metrics. Compare the moving average of true class probability of the selected samples across training epochs.

8.4 Scores computed from Training Dynamics of Clean and Noisy Labels

In this section, we study the scores to build a data selection curriculum for noisy-label learning (NLL). The data selection in NLL aims at addressing two major challenges, i.e., the detection of correct labels and the correction of wrong labels by pseudo labels. In the following, we will develop two score functions respectively for the two challenges, both built upon the training dynamics on individual samples in the process of NLL.

8.4.1 Loss Dynamics and Clean Label Detection

A key challenge for most noise-label learning methods is to design a reliable criterion to select/reweigh clean data and distinguish them from the noisy data, so all the clean data can be fully exploited while most noisy labels are filtered out of the training process. Loss computed at an instantaneous step have been widely used for this purpose according to the observation that the loss

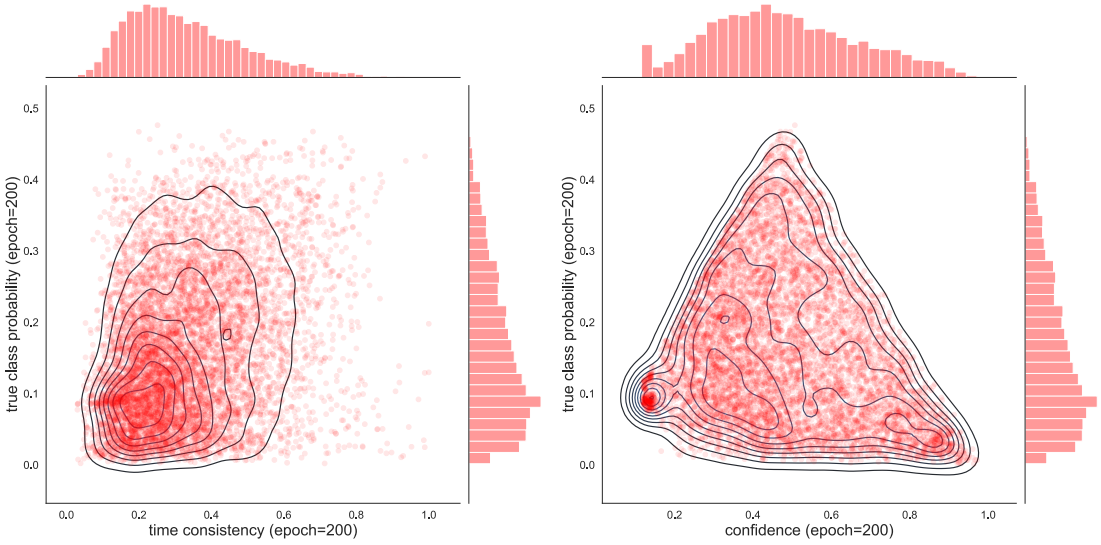

Figure 8.19: Compute time-consistency and confidence at epoch 200. Plot the incorrectly predicted samples in the validation set as a scatter plot with the selection metric as the x-axis and the true class probability as the y-axis.

on clean data is usually smaller than noisy data. One important reason is that the clean labels are mutually consistent with each other in producing gradient updates, and therefore, the model can fit them better and faster. On the other hand, the noisy labels may contain mutually inconsistent information, creating a form of long-lasting "tug of war" amongst themselves. For example, it can be hard for the model to find consistent visual patterns from images with noisy labels to make the desired predictions. However, instantaneous loss suffers from high variance across training epochs (as shown in the first plot of Figure 8.20) and is inaccurate for clean data detection under high noise ratio (i.e., the proportion of wrong labels is high) and the randomness of DNN training, e.g., random initialization, random data augmentation, etc. Moreover, it needs to evaluate the instantaneous loss for all samples in each step, resulting in extra inference cost on unselected samples.

Section 8.1 studies the dynamic patterns of losses over the course of training and gives us a new insight for better clean data detection even when the noise ratio (proportion of wrong labels) is high. In particular, we hypothesize that a sample's label is more likely to be correct if its losses persistently retain low values over training steps. Given a sample (x_i, y_i) with x_i being the features and y_i being the label, we describe its loss dynamics using a simple exponential moving average (EMA) of the instantaneous loss $\ell(f(x_i;\theta_t), y_i)$ (where $f(x_i;\theta_t)$ denotes the model output and θ_t is the model parameters at step t [‡] along the training history, which is defined and computed recursively as

$$l_{t+1}(i) = \begin{cases} \gamma \times \ell(f(x_i;\theta_t), y_i) + (1-\gamma) \times l_t(i) & \text{if } i \in S_t \\ l_t(i) & \text{else ,} \end{cases} \tag{8.24}$$

Where $\gamma \in [0, 1]$ is a discounting factor, V is the set of all n training samples, and $S_t \subseteq V$ is the set of samples selected (by a certain curriculum) for training at epoch t. We only update the EMA loss for selected samples using the byproduct $l_t(i)$ of training without requiring extra inference.

[‡]Each step does not strictly refer to a mini-batch: when the subset S_t for step t is larger than the mini-batch size (which is common in practice), we run SGD steps over all mini-batches of the subset for one pass.

In order to study the training dynamics over the course of noisy-label learning, we introduce a simple algorithm called "RoCL$_{Base}$" that alternates between supervised learning on given labels and self-supervision on pseudo labels. Its detailed procedure is listed in Algorithm 1.

Algorithm 1 RoCL$_{Base}$ (no curriculum)

1: **input:** $\{(x_i, y_i)\}_{i=1}^n$, $h(\cdot;\eta)$, $\ell(\cdot,\cdot)$, $f(\cdot;\theta)$, $T_{0:K}$; $\gamma \in [0,1]$
2: **initialize:** θ_0, $l_0(i) = c_0(i) = 0\ \forall i \in [n], T_{-1} = 0$
3: **for** $k \in \{0, \cdots, K\}$ **do**
4: **for** $t' \in \{1, \cdots, T_k\}$ **do**
5: $t \leftarrow t' + T_{k-1}$;
6: $S_t \leftarrow [n]$;
7: **if** $k\%2 = 0$ **then**
8: $\theta_t \leftarrow \theta_{t-1} + h\left(\nabla_\theta \frac{1}{n} \sum_{i=0}^n \ell_t(i);\eta\right)$; {supervised learning using given labels}
9: Update $l_{t+1}(i)$ by Eq. (8.24); {update EMA loss}
10: **else**
11: $\theta_t \leftarrow \theta_{t-1} + h\left(\nabla_\theta \frac{1}{n} \sum_{i=0}^n \zeta_t(i);\eta\right)$; {self-supervised learning using pseudo labels}
12: Update $c_{t+1}(i)$ by Eq. (8.26); {update EMA consistency loss}
13: **end if**
14: **end for**
15: **end for**

Note the EMA metrics in line 9 and line 12 are not used for training in RoCL$_{Base}$. They have been updated and recorded for the purpose of empirical study presented in the following.

In the first and the third plots of Figure 8.20, we show how the losses and EMA losses associated with clean/noisy data change throughout the training process. Specifically, we train a ResNet34 [81] model on CIFAR10 with 60% of the original labels randomly changed to a wrong class. To avoid quick overfitting to the noise, we train the model for multiple episodes (each composed of several epochs over all data with a cosine annealing learning rate) and alternate between the supervised learning episode that minimizes the cross-entropy loss against the given noisy labels and the self-supervision episode that minimizes the consistency loss Eq.(8.25) against the pseudo labels. Comparing the shaded areas (std) of instantaneous loss and EMA loss and the gap between curves in the two plots, we see that EMA leads to smaller variance within each group and larger gap between

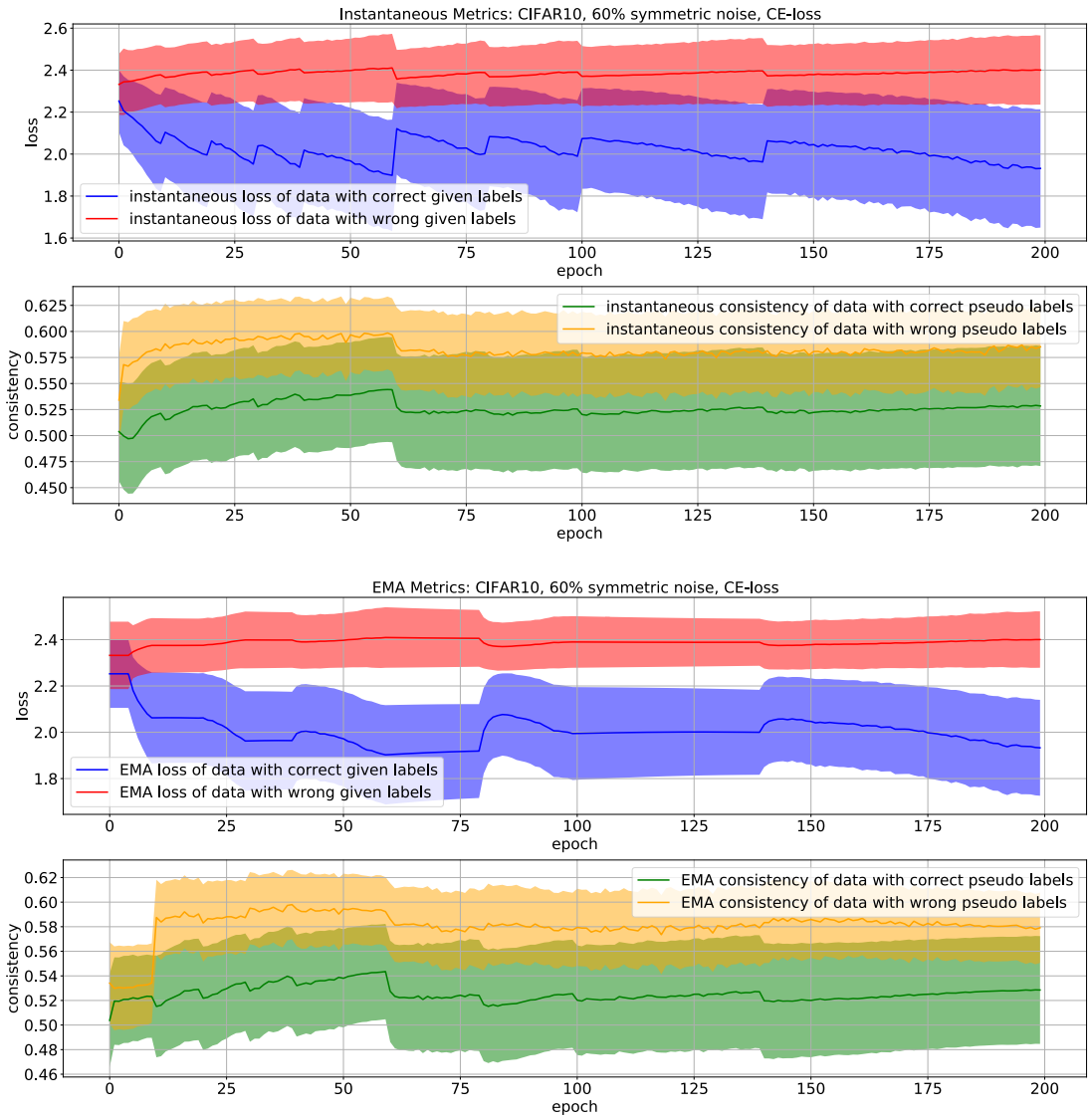

Figure 8.20: Dynamic patterns (mean±std) of instantaneous metrics (top) and exponential moving average (EMA) metrics (bottom) when applying Algorithm 1 that alternates between supervised learning on given labels and self-supervision on pseudo labels. *Larger gap between curves in each plot is better.* Symmetric noise is defined in the beginning of Section 16.2.2. We use cross entropy for supervised loss $\ell(\cdot, \cdot)$ in Eq. (8.24) and 0-1 loss for $\ell(\cdot, \cdot)$ in consistency loss Eq. (8.25) with $m = 7$.

the clean and noisy groups, demonstrating the effectiveness of EMA loss for clean data detection.

8.4.2 Consistency Dynamics and Pseudo Label Selection

Simply removing noisy data (x, y) with wrong label y discards important information about the data distribution $p(x)$ [4]. Including all the noisy data for self-supervision, bootstrapping or relabeling can also harm the training since the pseudo labels' quality is not equal across samples and much depends on the model generating them, where the model's predictions may contain errors that can accumulate if adopted for training. Hence, a careful selection of noisy data is necessary. However, without access to a purely clean dataset or a reliable pre-trained model, it is nontrivial to evaluate pseudo labels' correctness. By analyzing the training dynamics of model outputs in the above experiments, we discover that the model output for a sample tends to be an accurate pseudo label if the output remains consistent over training steps and across different augmentations of the sample. Such a consistency reflects that the output is consistent with the output of nearby augmentations of the same sample and has less conflict with other samples—otherwise the output would sensitively changes across steps training with different mini-batches of data. For an incorrect output, in contrast, the consistency tends to be weak since each of its augmentation may have different conflicts with different mini-batch of data over training steps.

We here define the instantaneous consistency loss of a sample x_i at step t as the discrepancy of the model output $f(x_i; \theta)$ between step t and $t-1$ on x_i and its m data augmentations $\{x_i^{(j)}\}_{j=1}^m$, where the discrepancy can be measured by any loss function $\ell(\cdot, \cdot)$. However, the discrepancy can be small if the models of epoch t and $t-1$ are too similar and make the same errors. Therefore, we use an exponential moving average of the model parameters (according to mean teacher [209]) and compute the prediction at step $t-1$ by averaging over multiple data augmentations (according to MixMatch [22]): $\overline{f}_t(x_i) \triangleq {}^1\!/_m \sum_{j=1}^m f(x_i^{(j)}; \overline{\theta}_t)$, $\overline{\theta}_t \triangleq \gamma\theta_{t-1} + (1-\gamma)\overline{\theta}_{t-1}$. Computing pseudo labels on augmented data and a time averaging ensemble of models is commonly-adopted for

semi-supervised learning [182, 117, 251]. The instantaneous consistency loss[§] $\zeta_t(i)$ of sample x_i at step t is then defined as

$$\zeta_t(i) \triangleq \frac{1}{m}\sum_{j=1}^{m} \ell(f(x_i;\theta_t), \overline{f}_t(x_i^{(j)})), \tag{8.25}$$

Which can also be minimized as a consistency loss for x_i in self-supervised learning when no label is given. Similar to the EMA loss in Eq. (8.24), we define the EMA consistency loss over training history

$$c_{t+1}(i) = \begin{cases} \gamma \times \zeta_t(i) + (1-\gamma) \times c_t(i) & \text{if } i \in S_t \\ c_t(i) & \text{else.} \end{cases} \tag{8.26}$$

Note we use the same γ value as in Eq. (8.24). The EMA consistency loss $c_t(i)$ measures both the time-consistency in Section 8.3 over multiple training steps and the spatial consistency over different augmentations of sample x_i. If the output prediction is wrong and contradicts other samples' labels, it will be inconsistent over time and across augmentations since it can easily change or flip after the next training step. In the last plot of Figure 8.20, we report the mean and standard deviation (the middle line and the shaded area) of the EMA consistency loss for two groups of data at each epoch, i.e., the ones with correct pseudo labels and the ones with incorrect pseudo labels. Comparing to the instantaneous consistency loss in the second plot, EMA consistency loss is a more reliable criterion for allocating correct pseudo labels. Thus, we can safely learn the noisy data by using their pseudo labels as training targets and avoid introducing harmful noises.

[§]This name maybe misleading since it actually measures the "inconsistency", but it is commonly called "consistency" in previous works [258, 22], so we decide to use the same name.

Part III

ALGORITHMS FOR CURRICULUM LEARNING

In this part, we propose several practical algorithms of curriculum learning, which are developed based on the problem formulations and score functions introduced in Part I and Part II. We will mainly discuss (1) optimization methods for the problem formulations; (2) planning and scheduling of different learning stages in the curricula; and (3) the selection criteria developed based on the score functions. The proposed algorithms cover a broad class of machine learning tasks including supervised learning, semi-supervised learning, self-supervised learning, and noisy-label learning. Moreover, we analyze the theoretical properties and/or empirical characteristics of the proposed methods in order to provide in-depth understandings of the proposed algorithms. In addition, we introduce several practical modifications and heuristic improvements to the proposed algorithms, whose effectiveness has been thoroughly verified in our experiments presented in Part IV.

Chapter 9

MINIMAX CURRICULUM LEARNING (MCL)

In order to overcome the drawbacks of the min-min formulation in Section 2.1, we introduce a minimax formulation of curriculum learning in Section 2.2 that chooses the hardest diverse samples. This formulation leads to the minimax curriculum learning (MCL) algorithm. MCL's minimax formulation is different from the min-min formulation used in SPL/SPLD. For certain losses and models, $L(y_i, f(x_i, w))$ is convex in w. The min-min formulation, however, is only bi-convex and requires procedures such as alternative convex search (ACS) as in [14]. Furthermore, diversity regularization of ν in SPLD leads to the loss of bi-convexity altogether.

Minimizing the worst case loss, as in MCL, is a widely used strategy in machine learning [119, 60, 189] to achieve better generalization performance and model robustness, especially when strong assumptions cannot be made about the data distribution. Compared to SPL/SPLD, MCL is also better in that the outer minimization over w in Eq. (2.2) is a convex program, and corresponds to minimizing the objective $g(w)$ in Eq. (9.1). On the other hand, querying $g(w)$ requires submodular maximization which can only be solved approximately.

In this chapter, we develop a planning strategy for MCL that firstly focuses on a few diverse samples in the early training stages and then gradually increases both the amount and the hardness of the training samples for later stages. We resort this curriculum to solving a sequence of continuous-discrete minimax optimization problems in the form of Eq. (2.2) associated with an increasing subset size k and a decreasing diversity weight λ. In order to solve them efficiently, we develop a provable algorithm to (approximately) for MCL, which alternates between data selection and model training.

9.1 Planning and Scheduling of Learning Stages in MCL

MCL's objective in Eq. (2.2) is composed of a hardness score (measured by summing the loss over all selected samples) and a diversity score of the selected subset. By annealing the weight λ for the diversity score from a large value, the early rounds of MCL mainly choose diverse samples with higher priority than hard samples. It then actually *decreases*, rather than increases, diversity as training rounds proceed. Our contention is that diversity is more important during the early phases of training when only relatively few samples are selected. Later rounds of training will naturally have more diversity opportunity simply because the size of the selected samples is much larger. Also, to avoid successive rounds selecting similar sets of samples, our approach selects the hardest, rather than the easiest, samples at each round. Hence, if a set of samples is learnt well during one training round, those samples will tend to be ill-favored in the next round because they become easier. We also measure hardness via the loss function, but the selection is always based on the hardest and most diverse samples of a given size k, where the degree of diversity is controlled by a parameter λ, and where diversity is measured by an arbitrary non-monotone submodular function. In fact, for binary variables the group sparse norm is also submodular where $\|\nu\|_{2,1} = \sum_{j=1}^{b} \sqrt{|C_j \cap A|} = F(A)$ where A is the set for which ν is the characteristic vector (i.e., $\nu_i = 1$ if $i \in A$ else $\nu_i = 0$), and C_j is the set of samples in the j^{th} group. Our approach allows the full expressive class of submodular functions to be used to measure diversity since the selection phases is based on submodular optimization.

Evidence for the naturalness of such hardness and diversity adjustment in a curriculum can also be found in human education. For example, courses in primary school usually cover a broad range of topics with highly selected contents, in order to expose the young learner efficiently to a diversity of knowledge early on. In college and graduate school, by contrast, students focus on advanced deeper knowledge within their majors that are more challenging. As another example, studies of bilingualism [23, 128, 150, 112] show that learning multiple languages in childhood is

beneficial for future brain development, but early-age multi-lingual learning is usually not advanced or concentrated linguistically for any of the languages involved. Still other studies argue that difficulty can be desired at early human learning stages [26, 149].

The minimax problem in Eq. (2.2) can be seen as a two-person zero-sum game between a teacher (the maximizer) and a student (the minimizer): the teacher chooses training set A based on the student's feedback about the hardness (i.e., the loss achieved by current model w) and how diverse according to the teacher ($\lambda F(A)$), while the student updates w to reduce the loss on training set A (i.e., learn A) given by the teacher. Similar teacher-student interaction also exist in real life. In addition, the teacher usually introduces concepts at the beginning and asks a small number of easy questions from a diverse range of topics and receives feedback from the student, and then further trains the student on the topics the student finds difficult while eschewing topics the student has mastered.

9.2 Minimax Curriculum Learning Algorithm

The minimax problem in Eq. (2.2) can be written as $\min_{w\in\mathbb{R}^m} g(w)$, which minimizes the following objective $g(w)$.

$$g(w) \triangleq \max_{A\subseteq V, |A|\leq k} \sum_{i\in A} L\left(y_i, f(x_i, w)\right) + \lambda F(A) \tag{9.1}$$

If the loss function $L(y_i, f(x_i, w))$ is convex w.r.t. w, then $g(w)$ is convex but, as mentioned above, enumerating all subsets is intractable. Defining the discrete objective $G_w : 2^V \rightarrow \mathbb{R}_+$ where

$$G_w(A) \triangleq \sum_{i\in A} L\left(y_i, f(x_i, w)\right) + \lambda F(A). \tag{9.2}$$

shows that computing $g(w)$ in involves a discrete optimization over $G_w(A)$, a problem that is submodular since $G_w(A)$ is weighted sum of a non-negative (since loss is non-negative) modular and a submodular function, and thus G_w is monotone non-decreasing submodular. Thus, the fast

greedy procedure mentioned earlier can be used to approximately optimizes $G_w(A)$ for any w.

Let $\hat{A}_w \subseteq V$ be the k-constrained greedy approximation to maximizing $G_w(A)$. We define the following approximate objective:

$$\hat{g}(w) \triangleq \sum_{i \in \hat{A}_w} L\left(y_i, f(x_i, w)\right) + \lambda F(\hat{A}), \tag{9.3}$$

and note that it satisfies $\alpha g(w) \leq \hat{g}(w) \leq g(w)$ where α is the approximation factor of submodular optimization. For $\tilde{w}$ within a region around w, $\hat{g}(\tilde{w})$ will utilize the same set $\hat{A}_w$. Therefore, $\hat{g}(w)$ is piecewise convex, if the loss function $L(y_i, f(x_i, w))$ is convex w.r.t. w, and different regions of within $\mathbb{R}^m$ are associated with different $\hat{A}$ although not necessarily the same regions or sets that define $g(w)$. We show in Section 9.4 that minimizing $\hat{g}(w)$ offers an approximate solution to Eq. (2.2).

With $\hat{g}(w)$ given, our algorithm is simply gradient descent for minimizing $\hat{g}(w)$, where many off-the-shelf methods can be invoked, e.g., SGD, momentum methods, Nesterov's accelerated gradient [160], Adagrad [55], etc. The key problem is how to obtain $\hat{g}(w)$, which depends on suboptimal solutions in different regions of w. It is not necessary, however, to run submodular maximization for every region of w. Since we use gradient descent, we only need to know $\hat{g}(w)$ for w on the optimization path. At the beginning of each iteration, we fix w and use submodular maximization to achieve the $\hat{A}_w$ that defines $\hat{g}(w)$. Then a gradient update step is applied to $\hat{g}(w)$. Let A^*_w represent the optimal solution to Eq. (9.2), then $\hat{A}_w$ satisfies $G(\hat{A}) \geq \alpha G(A^*)$.

Algorithm 2 details MCL. Lines 5-10 solve the optimization in Eq. (2.2) with λ and k scheduled in line 11. Lines 6-7 finds an approximate $\hat{A}$ via submodular maximization, discussed further in Section 9.3. Lines 8-9 update w for the current $\hat{A}$ by gradient descent $\pi(\cdot, \eta)$ with learning rate η. The inner optimization stops after p steps and then λ is reduced by factor $1 - \gamma$ where $\gamma \in [0, 1]$ and k is increased by Δ. The outer optimization stops after T steps when a form of "convergence",

described below, is achieved. Given $\hat{A}_w$, $\hat{g}(w)$ has gradient

$$\nabla \hat{g}(w) = \frac{\partial}{\partial w} \sum_{i \in \hat{A}_w} L\left(y_i, f(x_i, w)\right), \tag{9.4}$$

and thus gradient descent method can update w. For example, we can treat $\hat{A}$ as a batch if k is small, and update w by $w \leftarrow w - \eta \nabla \hat{g}(w)$ with learning rate η. For large $\hat{A}_w$, we can use SGD that applies an update rule to mini-batches within $\hat{A}_w$. More complex gradient descent rules $\pi(\cdot, \eta)$ can take historical gradients and w_τ^t's into account leading to $w^{t+1} \leftarrow w^t + \pi\left(\{w^{1:t}\}, \{\nabla \hat{g}(w^{1:t})\}, \eta\right)$.

Algorithm 2 Minimax Curriculum Learning (MCL)

input: $\pi(\cdot, \eta)$, γ, p, Δ,$\tilde{\alpha}$
output: w_T^0
initialize: $\tau \leftarrow 1$, w_τ^0, λ, k,
while *not "converged"* **do**
  for $t \in \{0, \cdots, p\}$ **do**
    $G(A) \leftarrow \sum_{i \in A} L\left(y_i, f(x_i, w_\tau^t)\right) + \lambda F(A)$;
    $\hat{A} \leftarrow$ WS-SUBMODULARMAX$(G, k, \hat{A}, \tilde{\alpha})$;
    $\nabla \hat{g}(w_\tau^t) = \frac{\partial}{\partial w} \sum_{i \in \hat{A}} L\left(y_i, f(x_i, w_\tau^t)\right)$;
    $w_\tau^{t+1} \leftarrow w_\tau^t + \pi\left(\{w_\tau^{1:t}\}, \{\nabla \hat{g}(w_\tau^{1:t})\}, \eta\right)$;
  end for
  $w_{\tau+1}^0 \leftarrow w_\tau^p$, $\lambda \leftarrow (1-\gamma) \cdot \lambda$, $k \leftarrow k + \Delta$, $\tau \leftarrow \tau + 1$;
end while

Considering the outer loop as well, the algorithm approximately solves a sequence of Eq. (2.2)s with decreasing λ and increasing k, where the previous solutions act as a warm start for the next iterations. This corresponds to repeatedly updating the model w on a sequence of training sets $\hat{A}$ that changes from small, diverse, and hard to large.

9.3 Submodular Maximization as a Subroutine

Although solving Eq. (9.2) exactly is NP-hard, a near-optimal solution can be achieved by the greedy algorithm, which offers a worst-case approximation factor of $\alpha = 1 - e^{-1}$ [158]. The algorithm starts with $A \leftarrow \emptyset$, and selects next the element with the largest marginal gain $f(v|A)$ from $V \backslash A$, i.e., $A \leftarrow A \cup \{v^*\}$ where $v^* \in \operatorname{argmax}_{v \in V \backslash A} f(v|A)$, and this repeats until $|A| = k$. It is simple to implement, fast, and usually outperforms other methods, e.g., those based on integer

linear programming. It requires $\mathcal{O}(nk)$ function evaluations for ground set size $|V| = n$. Since Algorithm 2 runs greedy Tp times, it is useful for the greedy procedure to be as fast as possible. The accelerated, or lazy, greedy algorithm [152] reduces the number of evaluations per step by updating a priority queue of marginal gains, while having the same output and guarantee as the original (thanks to submodularity) and offers significant speedups. Still faster variants are also available [153, 154]. Our own implementation takes advantage of the fact that line 7 of Algorithm 2 repeatedly solves submodular maximization over a sequence of submodular functions that are changing only slowly, and hence the previous set solution can be used as a warm start for the current algorithm, a process we call WS-SUBMODULARMAX outlined in Algorithm 3.

The greedy procedure offers much better approximation factors than $1 - e^{-1}$ when the objective $G(A)$ is close to modular. Specifically, the approximation factor becomes $\alpha = (1 - e^{-\kappa_G})/\kappa_G$ [41], which depends on the curvature $\kappa_G \in [0, 1]$ of $G(A)$ defined in Eq. 5.4. When $\kappa_G = 0$, G is modular, and when $\kappa_G = 1$, G is fully curved and the above bound recovers $1 - e^{-1}$. $G(A)$ becomes more modular as the outer loop proceeds since λ decreases. Therefore, the approximation improves with the number of outer loops. This has been proved in Lemma 1.

In MCL, therefore, the submodular approximation improves ($\alpha \to 1$) as λ grows, and the surrogate function $\hat{g}(w)$ correspondingly approaches the true convex objective $g(w)$.

9.3.1 Submodular Maximization Starting from a Previous "Warm" Solution

Algorithm 2 repeatedly runs a greedy procedure to solve submodular maximization, and this occurs two nested loops deep. In this section we describe how we speed this process up.

Our first strategy reduces the size of the ground set before starting a more expensive submodular maximization procedure. We use a method described in [229], which sorts the elements of V non-increasingly by $G(i|V \setminus i)$ and then remove any element i from V having $G(i) < G(\delta(k)|V \setminus \delta(k))$ where $\delta(k)$ is k^{th} element in the sorted permutation. Any such element will never be chosen by

the k-cardinality constrained greedy procedure because for any $\ell \in \{1, 2, \ldots, k\}$, and any set A, we have $G(\delta(\ell)|A) \geq G(\delta(\ell)|V \setminus \delta(\ell)) \geq G(\delta(k)|V \setminus \delta(k)) > G(i) \geq G(i|A)$ and thus greedy would always be able to choose an element better than i. This method results in no reduction in approximation quality. The speedup is data-dependent so the saved cost can be either large or small. But with a decreasing λ, $G(A)$ becomes more modular, and the filtering method can become more effective on pruning the ground set. It is worth noting that such pre-screening technique can be used together with existing fast greedy algorithms, e.g., those based on priority queue, which also becomes more efficient when $G(A)$ becomes more modular. Other methods we can employ are those such as [249, 153], resulting in small reduction in approximation quality, but we do not describe these further.

The key contribution of this section is a method exploiting a potential warm start set that might already achieve a sufficient approximation quality. Normally, the greedy procedure starts with the empty set and adds elements greedily until a set of size k is reached. In Algorithm 2, by contrast, a previous iteration has already solved a size-k constrained submodular maximization problem for the previous submodular function, the solution to which is one that could very nearly already satisfy a desired approximation bound for the current submodular function. The reason for this is that, depending on the weight update method in line 9 of Algorithm 2 between inner loop iterations, and the changes to parameters Δ and γ between outer iterations, the succession of submodular functions might not change very quickly. For example, when the learning rate η is small, the $\hat{A}$ from the previous iteration could still be valued highly by the current iteration's function, so running a greedy procedure from scratch is unnecessary. Our method warm-starts a submodular maximization process with a previously computed set, and offers a bound that trades off speed and approximation quality.

The approach is given in Algorithm 3, which (after the aforementioned filtering in line 3 [229]) tests in linear time if the warm start set $\hat{A}$ already achieves a sufficient approximation quality, and if

so, possibly improves it further with an additional linear or quasilinear time computation. To test approximation quality of $\hat{A}$, our approach uses a simple modular function upper bound, in line 4, to compute an upper bound on the global maximum value. For the subsequent improvement of $\hat{A}$, our approach utilizes a submodular semigradient approach [93] (specifically subgradients [66] in this case). If the warm-start set $\hat{A}$ does not achieve sufficient approximation quality in line 5, the algorithm backs off to standard submodular maximization in line 11 (we use the accelerated/lazy greedy procedure [152] here although other methods, e.g., [153], can be used as well).

Algorithm 3 Warm Start (WS) WS-SUBMODULARMAX$(G, k, \hat{A}, \tilde{\alpha} \in [0, 1))$

1: **Input:** $G(\cdot)$, k, $\hat{A}$, $\tilde{\alpha}$
2: **Output:** $\tilde{A}$
3: Reduce ground set size: arrange V non-increasingly in terms of $G(i|V\backslash i)$ in a permutation δ where $\delta(k)$ is the k^{th} element, set $V \leftarrow \{i \in V | G(i) \geq G(\delta(k)|V\backslash\delta(k))\}$;
4: Compute upper bound to maximum of Eq. (9.2):

$$\tau = \max_{A \in V, |A| \leq k} \sum_{i \in A} \left[L\left(y_i, f(x_i, w^t)\right) + \lambda F(i) \right]$$

5: **if** $G(\hat{A}) \geq \tilde{\alpha} \cdot \tau$ **then**
6: Permutation σ of V: the first k elements have $S^\sigma_k = \hat{A}$ and are chosen ordered non-increasing by $\kappa_G(v)$; the remaining $n - k$ elements $V \setminus \hat{A}$ for σ are chosen non-increasing by $\kappa_G(v)$.
7: Define modular function $h_{\hat{A}}(A) \triangleq \sum_{i \in A} h_{\hat{A}}(i)$ with $h_{\hat{A}}(\sigma(i)) = G(S^\sigma_i) - G(S^\sigma_{i-1})$;
8: Compute tight, at $\hat{A}$, lower bound $L(A)$ of $G(A)$:

$$L(A) \triangleq G(\hat{A}) + h_{\hat{A}}(A) - h_{\hat{A}}(\hat{A}) \leq G(A)$$

9: $\tilde{A} \leftarrow \text{argmax}_{A \in V, |A| \leq k} L(A)$;
10: **else**
11: $\tilde{A} \leftarrow$ LAZYGREEDY(G, V, k);
12: **end if**

Line 4 computes the upper bound $\tau \geq \max_{A \in V, |A| \leq k} G(A)$ which holds due to submodularity, requiring only a modular maximization problem (which can be done in $O(|V|)$ time, independent of k, to select the top k elements). Line 5 checks if an $\tilde{\alpha}$ approximation to this upper bound is achieved

by the warm-start set $\hat{A}$, and if not we back off to a standard submodular maximization procedure in line 11.

If $\hat{A}$ is an $\tilde{\alpha}$ approximation to the upper bound τ, then lines 6-9 runs a subgradient optimization procedure, a process that can potentially improve it further. The approach selects a subgradient defined by a permutation $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ of the elements. The algorithm then defines a modular function $L(A)$, tight at $\hat{A}$ and a lower bound everywhere else, i.e., $L(\hat{A}) = G(\hat{A})$, and $\forall A, L(A) \leq G(A)$. Any permutation will achieve this as long as $\hat{A} = \{\sigma(1), \sigma(2), \ldots, \sigma(k)\}$. The specific permutation we use is described below. Once we have the modular lower bound, we can do simple and fast modular maximization.

Lines 6-9 of Algorithm 3 offer a heuristic that can only improve the objective — letting $\tilde{A}$ be the solution after line 9, we have

$$G(\tilde{A}) \geq L(\tilde{A}) \geq L(\hat{A}) = G(\hat{A}). \tag{9.5}$$

The first inequality follows since $L(\cdot)$ is a lower bound of $G(\cdot)$; the second inequality follows from the optimality of $\hat{A}^+$; the equality follows since L is tight at $\hat{A}$.

The approximation factor $\tilde{\alpha}$ is distinct from the submodular maximization approximation factor α achieved by the greedy algorithm. Setting, for example $\tilde{\alpha} = 1 - 1/e$ would ask for the previous solution to be this good relative to τ, the upper bound on the global maximum, and the algorithm would almost always immediately jump to line 11 since achieving such approximation quality might not even be possible in polynomial time [61]. With $\tilde{\alpha}$ large, we recover the approximation factor of the greedy algorithm but ignore the warm start. If $\tilde{\alpha}$ is small, many iterations might use the warm start from the previous iteration, updating it only via one step of subgradient optimization, but with a worse approximation factor. In practice, therefore, we use a more lenient bound (often we set $\tilde{\alpha} = 1/2$) which is a good practical tradeoff between approximation accuracy and speed

(meaning lines 6-9 execute a reasonable fraction of the time leading to a good speedup, i.e., in our experiments, the time cost for WS-SUBMODULARMAX increases if $\alpha = 1$ by a factor ranging from about 3 to 5). In general, we have the following final bound based on the smaller of $\tilde{\alpha}$ and α.

Lemma 2. *Algorithm 3 outputs a solution $\hat{A}$ such that $G(\hat{A}) \geq \min\{\tilde{\alpha}, \alpha\} \times \max_{A \in V, |A| \leq k} G(A)$, where α is the approximation factor of the greedy procedure (typically $\alpha = (1 - e^{-\kappa_G})/\kappa_G$).*

Proof. Let A^* denote an optimal solution to Eq. (9.2):

$$A^* \in \operatorname*{argmax}_{A \in V, |A| \leq k} \sum_{i \in A} L\left(y_i, f(x_i, w^t)\right) + \lambda F(A). \tag{9.6}$$

τ computed in line 4 is an upper bound to $G(A^*)$ since:

$$\begin{aligned} \tau &\geq \textstyle\sum_{i \in A^*} \left[L\left(y_i, f(x_i, w^t)\right) + \lambda F(i)\right] \\ &= \textstyle\sum_{i \in A^*} L\left(y_i, f(x_i, w^t)\right) + \lambda \sum_{i \in A^*} F(i) \\ &\geq \textstyle\sum_{i \in A^*} L\left(y_i, f(x_i, w^t)\right) + \lambda F(A^*). \end{aligned} \tag{9.7}$$

The first inequality follows by the definition of τ; the last inequality is due to submodularity, guaranteeing $F(i) \geq F(i|B)$ for any $B \subseteq V$.

When $G(\hat{A}) \geq \tilde{\alpha} \cdot \tau$ (line 5), the subgradient ascent can only improve the objective. Thus, we have $G(\hat{A}) \geq \tilde{\alpha} \cdot \max_{A \in V, |A| \leq k} G(A)$ for $\hat{A}$ obtained in line 9. Otherwise, we run the greedy algorithm on the reduced ground set V. Thus, we have $G(\hat{A}) \geq \alpha \cdot \max_{A \in V, |A| \leq k} G(A)$ for $\hat{A}$ obtained in line 11. □

The heuristic in lines 6-9 is identical to one step of the semigradient-based minorization-maximization (MM) scheme used in, for example, [157, 96, 91, 93]. Which permutation to use for the subgradient in order to tighten the gap has been an issue discussed as far back as [157]. In the present work, we offer a heuristic for this problem. Let the first i elements in the permutation σ

be denoted $S_i^\sigma = \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}$, and let $A_{i-1}^\sigma \triangleq \{\sigma(j) \in A | j < i\} = S_{i-1}^\sigma \cap A \subseteq S_{i-1}^\sigma$ for any $i \in A$. The gap we wish to reduce is

$$0 \leq G(A) - L(A) = \sum_{\sigma(i) \in A} \left[G(\sigma(i)|A_{i-1}^\sigma) - G(\sigma(i)|S_{i-1}^\sigma) \right] \tag{9.8}$$

$$= \sum_{\sigma(i) \in A} G(\sigma(i)) \cdot \left[\frac{G(\sigma(i)|A_{i-1}^\sigma)}{G(\sigma(i))} - \frac{G(\sigma(i)|S_{i-1}^\sigma)}{G(\sigma(i))} \right] \tag{9.9}$$

$$\leq \sum_{\sigma(i) \in A} G(\sigma(i)) \cdot \left[\frac{G(\sigma(i)|A_{i-1}^\sigma)}{G(\sigma(i))} - (1 - \kappa_G(\sigma(i))) \right] \tag{9.10}$$

Which follows since $h_{\hat{A}}(\sigma(i)) = G(\sigma(i)|S_{i-1}^\sigma)$ by definition. Line 6 chooses a particular permutation in an attempt to reduce this gap. Define an element-wise form of curvature as $\kappa_G(v) = 1 - G(v|V \backslash v)/G(v) \in [0, 1]$ for all $v \in V$. Note that $\kappa_G = \max_{v \in V} \kappa_G(v)$. If $\kappa_G(v) \approx 0$ then G is practically modular at v and so $G(v|A) \approx G(v)$ for any set A; in other words, $G(v|A)$ is close to v's maximum possible gain even if v is ranked with index very late in the permutation σ where A is very large. If $\kappa_G(v) \approx 1$, on the other hand, then there is some set $A \subseteq V \setminus \{v\}$ that can appreciably reduce $G(v|A)$ relative to the maximum possible gain $G(v)$, and so it is best to rank v very early in the order σ where A must be a small set. One heuristic to achieve these goals is to choose a permutation σ that arranges the elements in an order non-increasing according to $\kappa_G(v)$, meaning $\kappa_G(\sigma(1)) \geq \kappa_G(\sigma(2)) \geq \ldots$. Choosing this order is therefore an attempt to keep each of the conditional gains $G(\sigma(i)|S_{i-1}^\sigma)$ as close as possible to $\sigma(i)$'s maximum possible gain, $G(\sigma(i))$. This corresponds to an attempt to reduce Eq. (9.9) (and correspondingly close the $G(A) - L(A)$ gap) as much as possible. Line 6 of Algorithm 3 does this, subject to the requirement that the first k elements of the permutation must correspond to $\hat{A}$ in order to be a subgradient. These tricks all help lines 6-9 produce a better updated approximate maximizer but at appreciably increased speed.

9.4 Conditions at Convergence

In this section, we study how close the solution $\hat{w}$ is of applying gradient descent to $\hat{g}(w)$, where we assume p is large enough so that a form of convergence occurs. Specifically, in Theorem 1, we analyze the upper bound on $\|\hat{w} - w^*\|_2^2$ based on two assumptions: 1) the loss $L\left(y_i, f(x_i, w)\right)$ being β-strongly convex w.r.t. w; and 2) $\hat{w}$ is achieved by running gradient descent in lines 6-9 of Algorithm 2 until convergence, defined as the gradient reaching zero. In case the loss $L\left(y_i, f(x_i, w)\right)$ is convex but not β-strongly convex, a commonly used trick to modify it to β-strongly convex is to add an ℓ_2 regularization $(\beta/2)\|w\|_2^2$. In addition, for non-convex $L\left(y_i, f(x_i, w)\right)$, it is possible to prove that with high probability, a noise perturbed SGD on $\hat{g}(w)$ can hit an ϵ-optimal local solution of $g(w)$ in polynomial time — we leave this for future work. In our empirical study (Section 15.1), MCL achieves good performance even when applied to non-convex deep neural networks. The following theorem relies on the fact that the maximum of multiple β-strongly convex functions is also β-strongly convex, shown in Appendix 9.6.1.

Theorem 1 (Inner-loop convergence)**.** *For the minimax problem in Eq. (2.2) with ground set of samples V and $\lambda \geq 0$, if the loss function $L\left(y_i, f(x_i, w)\right)$ is β-strongly convex and $|V| \geq k$, running lines 6-9 of Algorithm 2 until convergence (defined as the gradient reaching zero) yields a solution $\hat{w}$ satisfying*

$$\|\hat{w} - w^*\|_2^2 \leq \frac{2}{k\beta}\left(\frac{1}{\alpha} - 1\right) \cdot g(w^*), \tag{9.11}$$

$\hat{w}$ is the solution achieved at convergence, w^* is the optimal solution of the minimax problem in Eq. (2.2), $g(w^*)$ is the objective value achieved on w^*, and α is the approximation factor that submodular maximization can guarantee for $G(A)$.

The proof is given in Appendix 9.6.2.

It is interesting to note that the bound depends both on the strong convexity parameter β and on

the submodular maximization approximation α. As mentioned in Lemma 1, as λ gets smaller, the approximation factor α approaches 1 meaning that the bound in Equation (9.11) improves.

We mention the convergence criteria where the gradient reaches zero. While it is possible, in theory, for lines 6-9 of Algorithm 2 to oscillate amongst the non-differentiable boundaries between the convex pieces, with most damped learning rates, this will eventually subside and the algorithm will remain within one convex piece. The reason for this is line 7 of the algorithm always chooses one $\hat{A}$ thereby selecting one convex piece associated with the region around w_τ^t, and with only small subsequent adjustments to w_τ^t, the same $\hat{A}$ will continue to be selected. Hence, the algorithm will, in such case, reach the minimum of that convex piece where the gradient is zero.

We can restate and then simplify the above bound in terms of the resulting parameters, and corresponding λ, k values, used at a particular iteration τ of the outer loop. In the following, $\hat{w}_\tau$ is the solution achieved by Algorithm 2 at the iteration τ of the outer loop, and the optimal solution of the minimax problem in Eq.(2.2) with λ, k set as in iteration τ is denoted w_T^*.

Corollary 1. *If the loss function $L\left(y_i, f(x_i, w)\right)$ is β-strongly convex, the submodular function $F(\cdot)$ has curvature κ_F, and if each inner-loop in Algorithm 2 runs until convergence, then the solution $\hat{w}_\tau$ at the end of the τ^{th} iteration of the outer-loop fulfills:*

$$\|\hat{w}_\tau - w_\tau^*\|_2^2 \leq \frac{2\kappa_F}{k\beta(c_1/\lambda + 1)} g(w_\tau^*) \leq \frac{2\kappa_F}{\beta c_1} \times \frac{\lambda}{k} \times g(w_\tau^*), \tag{9.12}$$

where w_τ^* is the optimal solution of the minimax problem in Eq. (2.2) with λ set as in the τ^{th} outer loop iteration.

Thus, if k starts from k_0 and linearly increases via $k \leftarrow k+\Delta$ (as in line 11 of Algorithm alg:mcl),

$$\|\hat{w}_\tau - w_\tau^*\|_2^2 \leq \frac{2\kappa_F \lambda_0}{\beta c_1} \times \frac{(1-\gamma)^\tau}{(k_0 + \tau\Delta)} \times \left[g(w_\infty^*) + \lambda_0 c_2 (1-\gamma)^\tau\right], \tag{9.13}$$

Otherwise, if k increases exponentially, i.e., $k \leftarrow (1+\Delta) \cdot k$,

$$\|\hat{w}_\tau - w_\tau^*\|_2^2 \leq \frac{2\kappa_F \lambda_0}{\beta c_1 k_0} \times \left(\frac{1-\gamma}{1+\Delta}\right)^\tau \times \left[g(w_\infty^*) + \lambda_0 c_2 (1-\gamma)^\tau\right]. \tag{9.14}$$

In the above, λ_0 and k_0 are the initial values for λ and k, $c_1 = \min_{j \in V, t \in [1,\tau]} [L\left(y_i, f(x_i, \hat{w}_\tau^t)\right) / F(j)]$, $c_2 = \max_{A \subseteq V, |A| \leq k} F(A)$, and $g(w_\infty^*) = \min_{w \in \mathbb{R}^m} \max_{A \subseteq V, |A| \leq k} \sum_{i \in A} L\left(y_i, f(x_i, w)\right)$.

The proof can be found in Appendix 9.6.4. On the one hand, the upper bound above is in terms of the ratio λ/k which improves with larger subset sizes. On the other hand, submodular maximization becomes more expensive with k. Hence, Algorithm 2 chooses a schedule to decrease λ exponentially and increase k only linearly. Also, we see that the bound is dependent on the submodular curvature κ_F, the strongly-convex constant β, and c_1 which relates the submodular and modular terms (similar to as in Lemma 1). These quantities (κ_F/β and c_1) might be relevant for other convex-submodular optimization schemes.

9.5 Practical and Heuristic Improvements

There are several heuristic improvements we employ that are described next.

Algorithm 2 stops gradient descent after p steps. A reason for doing this is that $\hat{w}^p$ can be sufficient as a warm-start for the next iteration if p is large enough. We also have not observed any benefit for larger p, although we do eventually observe convergence empirically when the average loss no longer change appreciably between stages.

Also, lines 6-7 of Algorithm 2 require computing the loss on all the samples, and each step of the greedy algorithm needs to, in the worst case, evaluate the marginal gains of all of the unselected samples. Moreover, this is done repeatedly in the inner-most block of two nested loops. Therefore, we use two heuristic tricks to improve efficiency.

First, rather than selecting individual samples, we first cluster the data and then select clusters,

thereby reducing the ground set size from the number of samples to the number of clusters. We replace the per-sample loss $L\left(y_i, f(x_i, w)\right)$ with a per-cluster loss $L\left(Y^{(i)}, f(X^{(i)}, w)\right)$ that we approximate by the loss of the sample closest to the centroid within each cluster:

$$L\left(Y^{(i)}, f(X^{(i)}, w)\right) \triangleq \sum\nolimits_{j \in C^{(i)}} L\left(y_j, f(x_j, w)\right) \approx |C^{(i)}| L\left(y^{(i)}, f(x^{(i)}, w)\right), \tag{9.15}$$

where $C^{(i)}$ is the set of indices of the samples in the i^{th} cluster, and $x^{(i)}$ with label $y^{(i)}$ is the sample closest to the cluster centroid. We find that the loss on $x^{(i)}$ is sufficiently representative to approximately indicate the hardness of the cluster. The set V becomes the set of clusters and $A \subseteq V$ is a set of clusters, and hence the ground set size is reduced speeding up the greedy algorithm. When computing $F(A)$, the diversity of selected clusters, cluster centroids again represent the cluster. In line 8, the gradient is computed on all the samples in the selected clusters rather than on only $x^{(i)}$ at which point the labels of all the samples in the selected clusters are used. Otherwise, when selecting clusters via submodular maximization, the labels of only the centroid samples are needed. Thus, we need only annotate and compute the loss for samples in the selected clusters and the representative centroid samples $x^{(i)}$ of other clusters. This also reduces the need to label all samples up front as only the labels of the selected clusters, and centroid samples of each cluster, are used (i.e., the clustering process itself does not use the labels).

We can further reduce the ground set to save computation during submodular maximization via pre-filtering methods that lead either to no [229] or little [249, 153] reduction in approximation quality. Moreover, as λ decreases in the MCL objective and $G(A)$ becomes more modular, pruning method become more effective. More details are given in Section 9.3.1.

In Section 15.1, we evaluate the proposed MCL algorithm and its variants on multiple datasets. Moreover, we compare them with previous curriculum learning methods as well as the most widely used random mini-batch SGD. MCL performs the best among all the baselines and improves both

the training efficiency and the test accuracy.

9.6 Appendix

9.6.1 Proof of Proposition 1

Proposition 1. *The maximum of multiple β-strongly convex functions is β-strongly convex as well.*

Proof. Let $g(x) = \max_i g_i(x)$, where $g_i(x)$ is β-strongly convex for any i. According to a definition of strongly convex function given in Theorem 2.1.9 (page 64) of [159], $\forall \lambda \in [0, 1]$, we have

$$g_i(\lambda x + (1-\lambda)y) \leq \lambda g_i(x) + (1-\lambda)g_i(y) - \frac{\beta}{2}\lambda(1-\lambda)\|x-y\|_2^2, \forall i.$$

The following proves that $g(x)$ is also β-strongly convex:

$$\begin{aligned}
&g(\lambda x + (1-\lambda)y) = \max_i g_i(\lambda x + (1-\lambda)y) \\
\leq &\max_i \left[\lambda g_i(x) + (1-\lambda)g_i(y)\right] - \frac{\beta}{2}\lambda(1-\lambda)\|x-y\|_2^2 \\
\leq &\max_i \lambda g_i(x) + \max_i (1-\lambda)g_i(y) - \frac{\beta}{2}\lambda(1-\lambda)\|x-y\|_2^2 \\
= &\lambda g(x) + (1-\lambda)g(y) - \frac{\beta}{2}\lambda(1-\lambda)\|x-y\|_2^2.
\end{aligned}$$

□

9.6.2 Proof of Theorem 1

Proof. The objective $g(w)$ of the minimax problem in Eq. (2.2) after eliminating A is given in Eq. (9.1). Since $G(A)$ in Eq. (9.2) is monotone non-decreasing submodular, the optimal subset A when defining $g(w)$ in Eq. (9.1) always has size k if $|V| \geq k$. In addition, because the loss function $L\left(y_i, f(x_i, w)\right)$ is β-strongly convex, $g(w)$ in Eq. (9.1) is the maximum over multiple $k\beta$-strongly convex functions with different A. According to Proposition 1, $g(w)$ is also $k\beta$-strongly convex,

i.e.,

$$g(\hat{w}) \geq g(w^*) + \nabla g(w^*)^T(\hat{w} - w^*) + \frac{k\beta}{2}\|\hat{w} - w^*\|_2^2, \ \ \forall \nabla g(w^*) \in \partial g(w^*). \tag{9.16}$$

Since the convex function $g(w)$ achieves minimum on w^*, it is valid to substitute $\nabla g(w^*) = 0 \in \partial g(w^*)$ into Eq. (9.16). After rearrangement, we have

$$\|\hat{w} - w^*\|_2^2 \leq \frac{2}{k\beta}\left[g(\hat{w}) - g(w^*)\right]. \tag{9.17}$$

In the following, we will prove $g(w^*) \geq \alpha \cdot g(\hat{w})$, which together with Eq. (9.17) will lead to the final bound showing how close $\hat{w}$ is to w^*.

Note $\hat{g}(w)$ (Eq. (9.3)) is a piecewise function, each piece of which is convex and associated with different $\hat{A}$ achieved by a submodular maximization algorithm of approximation factor α. Since $\hat{A}$ is not guaranteed to be a global maxima, unlike $g(w)$, the whole $\hat{g}(w)$ cannot be written as the maximum of multiple convex functions and thus can be non-convex. Therefore, gradient descent in lines 6-9 of Algorithm 2 can lead to either: 1) $\hat{w}$ is a global minima of $\hat{g}(w)$; or 2) $\hat{w}$ is a local minima of $\hat{g}(w)$. Saddle points do not exist on $\hat{g}(w)$ because each piece of it is convex. We are also assuming other issues associated with the boundaries between convex pieces do not repeatedly occur.

1) When $\hat{w}$ is a global minima of $\hat{g}(w)$, we have

$$g(w^*) \geq \hat{g}(w^*) \geq \hat{g}(\hat{w}) \geq \alpha \cdot g(\hat{w}). \tag{9.18}$$

The first inequality is due to $g(\cdot) \geq \hat{g}(\cdot)$. The second inequality is due to the global optimality of $\hat{w}$. The third inequality is due to the approximation bound $\hat{g}(\cdot) \geq \alpha \cdot g(\cdot)$ guaranteed by the submodular maximization in Step 7 of Algorithm 2.

2) When $\hat{w}$ is a local minima of $\hat{g}(w)$, we have $\nabla\hat{g}(\hat{w}) = 0$. Let $h(w)$ be the piece of $\hat{g}(w)$ where $\hat{w}$ is located, then $\hat{w}$ has to be a global minima of $h(w)$ due to the convexity of $h(w)$. Let $\mathcal{A}$ denote the ground set of $\hat{A}$ on all pieces of $\hat{g}(w)$, we define an auxiliary convex function $\tilde{g}(w)$ as

$$\tilde{g}(w) \triangleq \max_{A\in\mathcal{A}} \sum_{i\in A} L\left(y_i, f(x_i, w)\right) + \lambda F(A). \tag{9.19}$$

It is convex because it is defined as the maximum of multiple convex function. So we have

$$\hat{g}(w) \leq \tilde{g}(w) \leq g(w), \forall w \in \mathbb{R}^m. \tag{9.20}$$

The first inequality is due to the definition of $\mathcal{A}$, and the second inequality is a result of $\mathcal{A} \subseteq 2^V$ by comparing $g(w)$ in Eq. (9.1) with $\tilde{g}(w)$ in Eq. (9.19). Let $\tilde{w}$ denote a global minima of $\tilde{g}(w)$, we have

$$g(w^*) \geq \tilde{g}(w^*) \geq \tilde{g}(\tilde{w}) \geq h(\tilde{w}) \geq h(\hat{w}) = \hat{g}(\hat{w}) \geq \alpha \cdot g(\hat{w}). \tag{9.21}$$

The first inequality is due to Eq. (9.20), the second inequality is due to the global optimality of $\tilde{w}$ on $\tilde{g}(w)$, the third inequality is due to the definition of $\tilde{g}(w)$ in Eq. (9.19) ($\tilde{g}(w)$ is the maximum of all pieces of $\hat{g}(w)$ and $h(w)$ is one piece of them), the fourth inequality is due to the global optimality of $\hat{w}$ on $h(w)$, the last inequality is due to the approximation bound $\hat{g}(\cdot) \geq \alpha \cdot g(\cdot)$ guaranteed by the submodular maximization in Step 7 of Algorithm 2.

Therefore, in both cases we have $g(w^*) \geq \alpha \cdot g(\hat{w})$. Applying it to Eq. (9.17) results in

$$\|\hat{w} - w^*\|_2^2 \leq \frac{2}{k\beta}\left(\frac{1}{\alpha} - 1\right) \cdot g(w^*). \tag{9.22}$$

□

9.6.3 *Proposition 2*

Proposition 2. *If $x \in [0, 1]$, the following inequality holds true.*

$$\frac{x}{1 - e^{-x}} - 1 \leq x. \tag{9.23}$$

Proof. Due to two inequalities $e^x \leq 1 + x + x^2/2$ for $x \leq 0$ and $1 - e^{-x} \geq x/2$ for $x \in [0, 1]$,

$$\frac{x}{1 - e^{-x}} - 1 = \frac{x - 1 + e^{-x}}{1 - e^{-x}} \leq \frac{x - 1 + (1 - x + x^2/2)}{x/2} = x. \tag{9.24}$$

□

9.6.4 *Proof of Corollary 1*

Proof. Applying the inequality in Proposition 2 and the approximation factor of lazy greedy $\alpha = (1 - e^{-\kappa_G})/\kappa_G$ to the right hand side of Eq. (9.11) from Theorem 1 yields

$$\begin{aligned} \|\hat{w} - w^*\|_2^2 &\leq \frac{2}{k\beta} \left(\frac{1}{\alpha} - 1 \right) \cdot g(w^*) \\ &= \frac{2}{k\beta} \left(\frac{\kappa_G}{1 - e^{-\kappa_G}} - 1 \right) \cdot g(w^*) \leq \frac{2\kappa_G}{k\beta} \cdot g(w^*), \end{aligned} \tag{9.25}$$

where κ_G is the curvature of submodular function $G(\cdot)$ defined in Eq. (9.2). Substituting the inequality about κ_G from Lemma 1 into Eq. (9.25) results in

$$\|\hat{w} - w^*\|_2^2 \leq \frac{2\kappa_F}{k\beta(c_1/\lambda + 1)} \leq \frac{2\kappa_F}{\beta c_1} \times \frac{\lambda}{k} \times g(w^*). \tag{9.26}$$

We use subscript as the index for iterations in the outer-loop, e.g., $\hat{w}_T$ is the model weights w after the T^{th} iteration of outer-loop. If we decrease λ exponentially from $\lambda = \lambda_0$ and increase k linearly

from $k = k_0$, as Step 11 in Algorithm 2, we have

$$\|\hat{w}_T - w_T^*\|_2^2 \leq \frac{2\kappa_F \lambda_0}{\beta c_1} \times \frac{(1-\gamma)^T}{(k_0 + T\Delta)} \times g(w_T^*), \tag{9.27}$$

According to the definition of $g(\cdot)$ in Eq. (9.1), for $g(w_T^*)$ we have

$$\begin{aligned} g(w_T^*) &= \min_{w \in \mathbb{R}^m} \max_{A \subseteq V, |A| \leq k} \sum_{i \in A} L\left(y_i, f(x_i, w)\right) + \lambda F(A) \\ &\leq \min_{w \in \mathbb{R}^m} \max_{A \subseteq V, |A| \leq k} \sum_{i \in A} L\left(y_i, f(x_i, w)\right) + \lambda_0 (1-\gamma)^T \max_{A \subseteq V, |A| \leq k} F(A) \\ &= g(w_\infty^*) + \lambda_0 (1-\gamma)^T c_2, \end{aligned} \tag{9.28}$$

where

$$g(w_\infty^*) \triangleq \min_{w \in \mathbb{R}^m} \max_{A \subseteq V, |A| \leq k} \sum_{i \in A} L\left(y_i, f(x_i, w)\right), c_2 \triangleq \max_{A \subseteq V, |A| \leq k} F(A). \tag{9.29}$$

Substituting Eq. (9.28) to Eq. (9.27) yields

$$\|\hat{w}_T - w_T^*\|_2^2 \leq \frac{2\kappa_F \lambda_0}{\beta c_1} \times \frac{(1-\gamma)^T}{(k_0 + T\Delta)} \times \left[g(w_\infty^*) + \lambda_0 c_2 (1-\gamma)^T\right], \tag{9.30}$$

If we can tolerate more expensive computational cost for running submodular maximization with larger budget k, and increase k exponentially, i.e., $k \leftarrow (1+\Delta) \cdot k$, we have

$$\|\hat{w}_T - w_T^*\|_2^2 \leq \frac{2\kappa_F \lambda_0}{\beta c_1 k_0} \times \left(\frac{1-\gamma}{1+\Delta}\right)^T \times \left[g(w_\infty^*) + \lambda_0 c_2 (1-\gamma)^T\right]. \tag{9.31}$$

This completes the proof. □

Chapter 10

DYNAMIC INSTANCE GUIDED HARDNESS CURRICULUM LEARNING (DIHCL)

In this chapter, we introduce a curriculum learning algorithm, "*DIH guided curriculum learning* (**DIHCL**)", that is naturally motivated by the properties of DIH score discussed in Section 8.1. DIHCL keeps training the model on those samples that have historically been hard since the model does not perform well on them. By contrast, it is safe to revisit easy samples (those with small DIH values) less frequently because the model is more likely to stay at those samples' minima. Hence, DIHCL helps a model focus on that which it finds difficult. This is similar to strategies that improve human learning, such as the Leitner system for spaced repetition [127]. This is also analogous to boosting [185] — in boosting, however, we average the instantaneous sample performance of multiple weak learners at the current time, while in DIHCL we average the instantaneous sample performance of one strong learner over the training history.

At each training step, DIHCL selects a subset of samples according to their DIH values, where the hard samples have higher probabilities of being selected relative to the easy samples. The model is updated by (stochastic) gradients computed on the selected samples. We then update the DIH of the selected samples by using their instantaneous hardness, a byproduct of back-propagation (since it needs to perform inference at first, e.g., a forward-propagation of a DNN). This significantly improves the efficiency of previous CL methods, which rely on extra inference steps to evaluate the instantaneous hardness of all the samples. Here, it is safe to update only the DIH of the selected samples since the unselected ones have smaller and decreasing DIH values (due to the observed properties of DIH) and thus keeping a stale DIH for them will not reduce their chance of being

selected in the future steps. As mentioned earlier, in the training of DNNs, the hardness ranking of each sample by DIH will quickly converge after a few training steps and remains consistent for future steps. Those early steps provide the opportunity for the necessary exploration to ensure that hardness ranking is via DIH is accurate. To improve the exploration efficiency, DIHCL sweeps through the entire training set for the first few epochs and then starts to select training samples by DIH-weighted subset (random) sampling, and we gradually decrease the subset size during training. We provide several options for weighted sampling, using different distributions, and we integrate subset diversity into the selection criteria as well when feasible. Empirically, we evaluate several variants of DIHCL and compare them against random mini-batch SGD as well as recent curriculum learning algorithms on 11 datasets. DIHCL shows an advantage over other baselines in terms both of time/sample efficiency and test set accuracy.

10.1 A "Free" Curriculum based on Dynamic Instance Hardness

We arrive at a curriculum learning strategy that selects harder samples to train the model at each step. And since only a subset of samples at each epoch are used to train on, and since DIH is updated only for the selected samples, we find that computation to achieve a given accuracy is reduced. We give a greedy version of DIHCL in Algorithm 4, where $\{(x_i, y_i)\}_{i=1}^n$ is the training data, $\pi(\cdot; \eta)$ is an optimization method such as SGD, $\eta_{1:T}$ are the T learning rates, and γ_k is the reduction factor for subset sizes k_t. DIHCL trains using more samples early on to produce an accurate initial estimate of $r_t(i)$. This is indicated by T_0, the number of warm start epochs over the whole training set. After iteration T_0, we gradually reduce the number of samples from $k_1 = n$ to k_t thereby focusing on the most difficult samples as training proceeds. At step t, we select subset $S_t \subseteq [n]$ with large $r_{t-1}(i)$ and then update the model by training on S_t. We then update $r_t(i)$ via Eq. (8.2).

Since the learning rate can change over different steps, and large learning rates mean greater

Algorithm 4 DIH Curriculum Learning (DIHCL-Greedy)

1: **input:** $\{(x_i, y_i)\}_{i=1}^n$, $\pi(\cdot;\eta)$, $\ell(\cdot,\cdot)$, $F(\cdot;w)$;
$\eta_{1:T}$; T, T_0; $\gamma, \gamma_k \in [0,1]$
2: **initialize:** w, η_1, $k_1 = n$, $r_0(i) = 1\ \forall i \in [n]$
3: **for** $t \in \{1, \cdots, T\}$ **do**
4: **if** $t \leq T_0$ **then**
5: $S_t \leftarrow [n]$;
6: **else**
7: Let $S_t = \text{argmax}_{S:|S|=k_t} \sum_{i \in S} r_t(i)$;
8: **end if**
9: Apply optimization $\pi(\cdot;\eta)$ to update model:

$$w_t \leftarrow w_{t-1} + \pi\left(\nabla_w \sum_{i \in S_t} \ell(y_i, F(x_i; w_{t-1})); \eta_t\right)$$

10: Compute normalized $a_t(i)$ for $i \in S_t$ using Eq. (10.1);
11: Update DIH $r_{t+1}(i)$ using Eq. (8.2);
12: $k_{t+1} \leftarrow \gamma_k \times k_t$;
13: **end for**

model change, we normalize $a_t(i)$ by the learning rate η_{t-1}*. Specifically, we apply one of the following depending on which form of $a_t(i)$ we are using (case (A), (B), or (C) above):

$$\begin{aligned}
&(A)\ \ a_t(i) \leftarrow \ell(y_i, F(x_i; w_{t-1}))/\eta_t, \\
&(B)\ \ a_t(i) \leftarrow |\ell(y_i, F(x_i; w_{t-1})) - \ell(y_i, F(x_i; w_{\tau_t(i)-1}))|/\textstyle\sum_{t'=\tau_t(i)}^{t} \eta_{t'}, \\
&(C)\ \ a_t(i) \leftarrow |\mathbb{1}[\hat{y}_i^t = y_i] - \mathbb{1}[\hat{y}_i^{t-1} = y_i]|/\textstyle\sum_{t'=\tau_t(i)}^{t} \eta_{t'},
\end{aligned} \tag{10.1}$$

where $\tau_t(i) < t-1$ indicates the most recent step before $t-1$ when i was selected. The T_0 warm start epochs and the schedule of decreasing k_t are necessary for early exploration since DIH is a running mean over a sample's dynamics and thus needs to revisit each sample to estimate its relative DIH position. A simple method to further reduce training time in early stages is to extract and use only a small and diverse subset of S_t. Inspired by MCL [248], after line 7, we reduce S_t to a subset of

*We use η_{t-1} instead of η_t because $a_t(i)$ is computed based on w_{t-1} before the weight update in step t.

size $k'_t = \gamma_{k'} k_t$ ($0 < \gamma_{k'} \leq 1$) by (approximately) solving the following submodular maximization.

$$\max_{S \subseteq S_t, |S| \leq k'_t} \sum_{i \in S} r_t(i) + \lambda_t G(S) \tag{10.2}$$

The function $G : 2^{S_t} \rightarrow \mathbb{R}_+$ can be any submodular function [66], and hence we can exploit fast greedy algorithms [158, 152, 153] to solve Eq. (10.2) with an approximation guarantee. We gradually reduce preference for diversity as training proceeds by reduce λ_t by a factor $0 \leq \gamma_\lambda \leq 1$ at each step.

10.2 Practical DIHCL with Weighted Sampling

In line 7 of Algorithm 4, we select S_t with the highest $r_{t-1}(i)$ values. In practice, we find adding randomness to the selection procedure gives better performance as (1) exploration on samples with small $r_t(i)$ are necessary to accurately estimate to $r_t(i)$, and (2) randomness of training samples is essential to achieve a good quality solution w for non-convex models such as DNNs. Instead of choosing the top k_t samples greedily and deterministically, we perform a randomized greedy procedure by sampling with probability $p_{t,i} \propto h(r_{t-1}(i))$, where $h(\cdot)$ is a monotone non-decreasing function, similar to [133, 141]. Hence, we still prefer data points with high DIH. An ideal choice of $h(\cdot)$ should balance between the exploration (under poorly estimated DIH values) and exploitation (when DIH is well estimated). We propose the following three sampling methods to replace line 7 of Algorithm 4, and give extensive evaluations in the experimental section.

DIHCL-Rand: Let $h(r_t(i)) = r_t(i)$. We sample data with probability proportional to DIH values.

DIHCL-Exp: We trade-off exploration and exploitation similarly to Exp3 [9], which samples based on the softmax value. We then reweigh the observation by the selection probability to encourage exploration:$h(r_t(i)) = \exp\left[\sqrt{2 \log n / n} \times r_t(i)\right]$, $a_t(i) \leftarrow a_t(i) / p_{t,i}$ $\forall i \in S_t$.

DIHCL-Beta: We utilize the idea of Thompson sampling [210] and use a Beta prior distribution to balance exploration and exploitation, i.e., $h(r_t(i)) \sim \text{Beta}(r_t(i), c - r_t(i))$, where c is a sufficiently large constant with $c \geq r_t(i)$, e.g., $c = 1$ when $a_t(i)$ is prediction flip. The Beta distribution encourages exploration when the difference between $r_t(i)$ and $c - r_t(i)$ is small.

In Section 15.2, we evaluate different variants of DIHCL and compare them with several curriculum learning approaches on 11 image classification datasets.

Chapter 11

DYNAMICS-OPTIMIZED CURRICULUM LEARNING (DOCL)

In Chapter 4, we introduce a novel formulation of curriculum learning that aims at optimizing the training dynamics in each learning stage by selecting a subset of important training samples. Specifically, the training samples are selected to achieve the greatest progress and fastest learning speed towards the ground-truth on all available samples. In Section 8.2, we derive score functions for both regression and classification tasks from the optimization formulation, where the scores are computed from samples' residual and linear temporal dynamics. As a result, the optimization reduces to selecting the samples with the highest scores.

In this chapter, we develop the algorithm of "Dynamics-optimized Curriculum Learning (DoCL)", which is mainly built upon the derived scores and seamlessly incorporates several other techniques for better performance and efficiency. DoCL encourages the model to focus on the samples at the learning frontier, i.e., those with large loss but fast learning speed. The scores in discrete time can be estimated via already-available byproducts of training, and thus require negligible extra compute. We discuss the properties and potential advantages of the proposed dynamics optimization via current deep learning theory and empirical studies. By integrating it with cyclical training of neural networks, DoCL selects the training set at each step by a weighted sampling based on the scores.

11.1 Three Data Selection Criteria using DoCL scores

Algorithm 5 Baselines (the three curricula) in Section 11.1

input: $\{(x_i, y_i)\}_{i=1}^n$, $\ell(\cdot,\cdot)$, $f(\cdot;\theta)$, $\{\eta_t\}_{t=0}^{T_\kappa}$, $\{T_i\}_{i=0}^{\kappa}$, k
initialize: θ, $T_{-1}=0$, $k=n$, $\rho_i=0, g_i=f(x_i)\ \ \forall i\in[n]$
for $j\in\{0,\cdots,\kappa\}$ **do**
  for $t\in\{T_{j-1},\cdots,T_j\}$ **do**
    if $j>0$ **then**
      Baseline1: Alternating between the highest and lowest scored samples:
      if $j\%2=1$ **then**
        $S_t\leftarrow$ top-k samples with the **largest** score $\hat{a}_t(i)$;
      else
        $S_t\leftarrow$ top-k samples with the **smallest** score $\hat{a}_t(i)$;
      end if
      Baseline2: always selecting the highest-scored samples:
      $S_t\leftarrow$ top-k samples with the **largest** score $\hat{a}_t(i)$;
      Baseline3: always selecting the lowest-scored samples:
      $S_t\leftarrow$ top-k samples with the **smallest** score $\hat{a}_t(i)$;
      Update θ by mini-batch SGD with learning rate η_t to minimize the task's loss $\ell(\cdot)$ on S_t;
    end if
    Estimate linear dynamics $\left.\frac{\partial f(x_i;\theta_t)}{\partial t}\right|_D$ *and compute scores* $\hat{a}_{t+1}(i)$:
    Uniform sampling $D\subseteq[n]$ up to size n;
    Update θ by large-batch SGD with learning rate η_t to minimize L2 loss on D;
    Compute $f(x_i)$ for all samples $i\in[n]$;
    for $i\in\{1,\cdots,n\}$ **do**
      $\rho_i\leftarrow\rho_i+\eta_t, \frac{\partial f(x_i)}{\partial t}=\frac{f(x_i)-g_i}{\rho_i}$;
      Restore $\rho_i\leftarrow 0$ and $g_i\leftarrow f(x_i)$;
      Compute $a_t(i)$ by Eq. (8.4) (regression) or Eq. (8.7) (classification);
      Update the exponential moving average $\hat{a}_{t+1}(i)$ for all samples $i\in[n]$ using Eq. (8.5);
    end for
  end for
end for

The analysis of dynamics-optimization in Section 8.2 implies that we should select samples with larger scores $\hat{a}_t(i)$ (Eq. (8.5)) for training in each step. In this section, we present an empirical study of the training dynamics with different data selection criteria in a more primitive framework, i.e., Algorithm 5, which in each step computes the DoCL score for all samples and then trains

the model on the top-k samples with the largest or the smallest scores. **The aim is to solely evaluate the effectiveness of the proposed scores and rule out influences of any additional heuristics, techniques, or hyperparameters** that we will introduce later for building a more practical algorithm (DoCL in Algorithm 6).

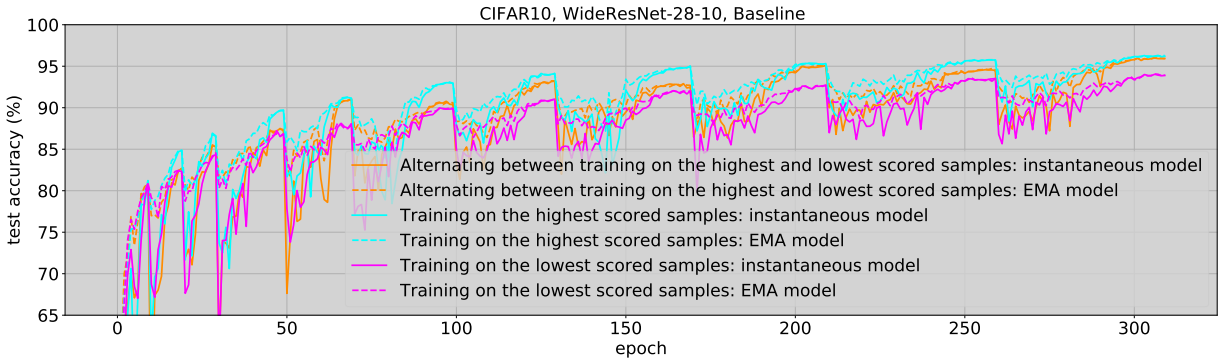

Figure 11.1: Test set accuracy of WideResNet-28-10 (instantaneous model) and its exponential moving average (EMA model) during the course of training when using the three data selection curricula.

In particular, we train a WideResNet-28-10 on CIFAR10 for multiple episodes/cycles each applying SGD with cosine annealing learning rates. In every epoch of an episode, we select $k = 10000$ samples to train the model and we compare three data selection curricula: (1) *Baseline1* selects the k highest-scored samples in oddly-numbered episodes and k lowest-scored samples in evenly-numbered episodes; (2) *Baseline2* always selects the k highest-scored samples; and (3) *Baseline3* always selects the k lowest-scored samples. To update the scores for all the n samples, in each epoch, we uniformly draw 2048 samples as D to estimate the linear dynamics $\frac{\partial f(x_i;\theta_t)}{\partial t}\Big|_D$ in Eq. (8.7), and we apply inference on all samples to obtain $f(x_i)$ (costly for practice usage).

We first compare the test set accuracy of the three curricula in Figure 11.1. Baseline2 keeps achieving the highest test accuracy among the three since very early episodes. In addition, Baseline1 and Baseline2 outperform Baseline3 by a large margin. This indicates that the samples with higher scores bring more improvement to the generalization performance than the ones with lower scores.

Next, we take a closer look at the proposed score $\hat{a}_t(i)$ on selected samples, the DoCL objective

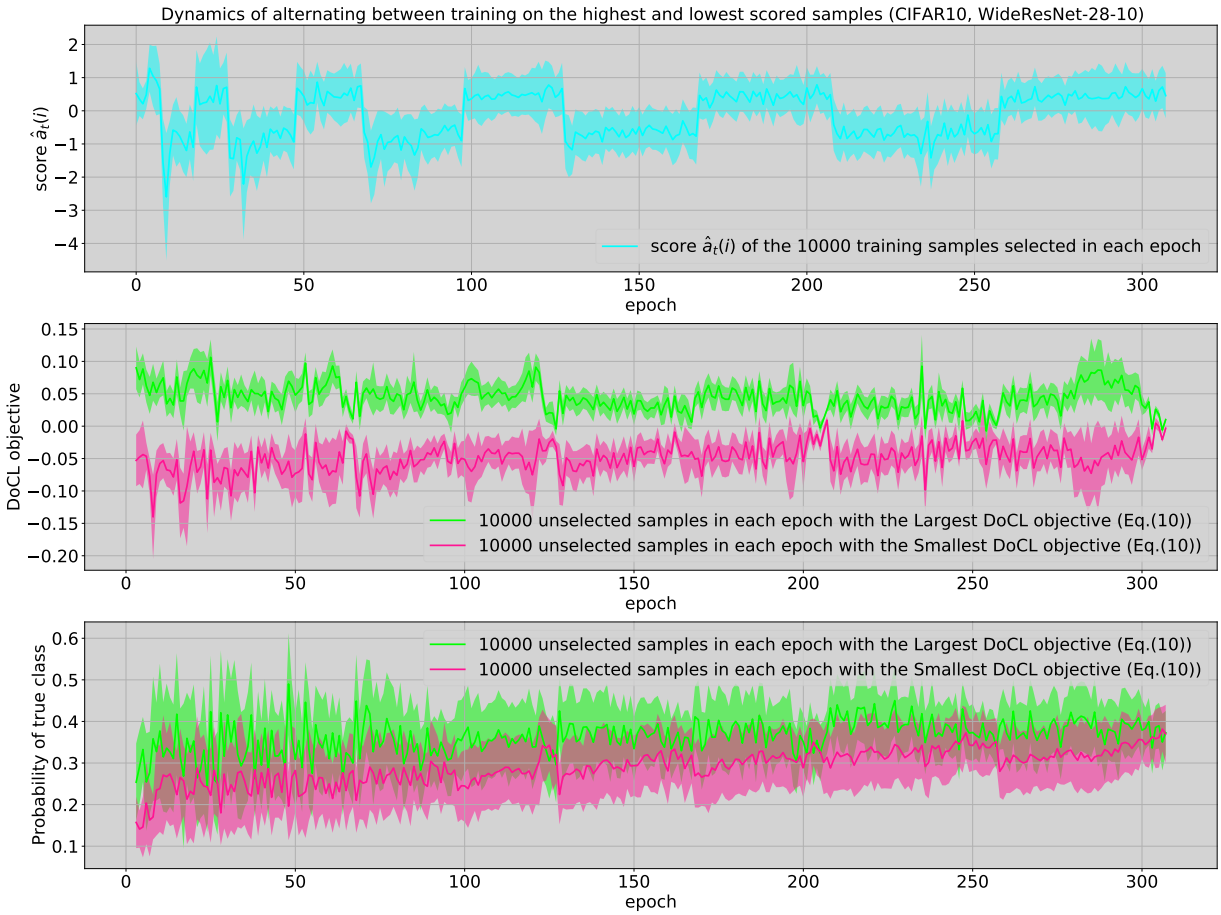

Figure 11.2: *Baseline1 alternates between the highest and lowest scored samples*: Dynamics (mean±std) for (**Top**) the score $\hat{a}(i)$ (Eq. (8.5)) of the selected samples, (**Middle**) DoCL objective (Eq. (4.12)) values of unselected samples, and (**Bottom**) output true-class probabilities for unselected samples. We split the unselected samples in each epoch into two groups with the largest/smallest DoCL objective values.

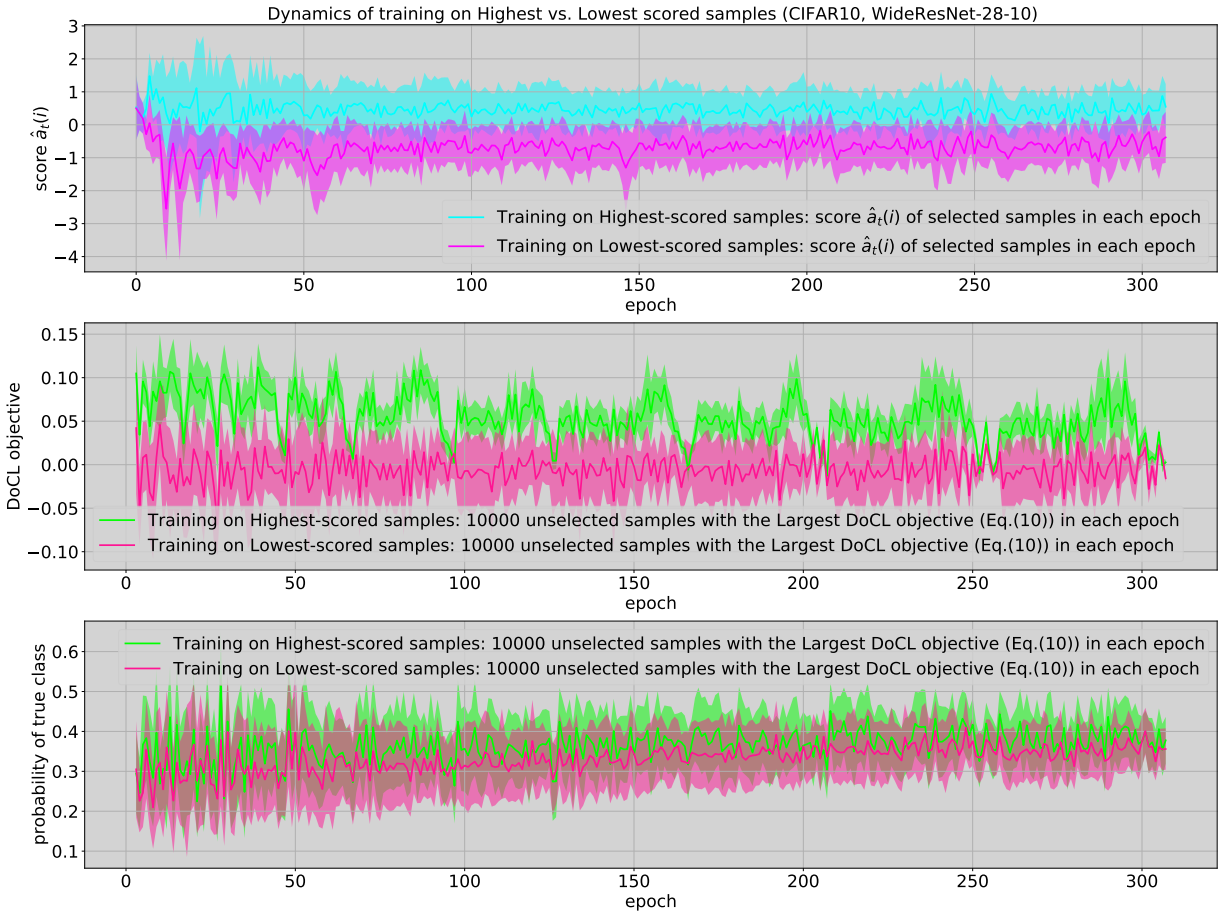

Figure 11.3: *Training with highest-scored (Baseline2) vs. lowest-scored (Baseline3) samples*: Dynamics (mean±std) for (**Top**) the score $\hat{a}(i)$ (Eq. (8.5)) of the selected samples, (**Middle**) DoCL objective (Eq. (4.12)) values and (**Bottom**) output true-class probabilities for unselected samples with the largest DoCL objective values.

(Eq. (4.12)), the prediction quality (measured by true class probabilities) of unselected data, and their correlations during the course of training. The results verify that optimizing the learning dynamics indeed improves the generalization performance and training on high-scored samples. In Figure 11.2, we report the dynamics observed on Baseline1. In the middle and bottom plots, we split the unselected samples in each epoch into two groups, i.e., the 10000 samples with the largest DoCL objective values and the 10000 samples with the smallest DoCL objective values. They together show that the model performs better (i.e., producing higher true-class probabilities) on samples with larger DoCL objective values. Hence, optimizing the learning dynamics of all samples (not only the selected training samples) is consistent with the learning goal of reducing the classification error over the data distribution. Moreover, in the top and middle plots, we observe that the DoCL objective (for both groups) degrades when training on the lowest-scored samples, while it increases when training on the highest-scored ones. It indicates that the highest-scored samples improve the DoCL objectives more effectively and result in better prediction qualities.

In Figure 11.3, we compare the dynamic patterns of Baseline2 and Baseline3. The top plot shows that the samples are distinguishable based on their scores, indicating that they are not equal in accelerating the learning process and thus a data selection curriculum can be better than uniform sampling. In the middle and bottom plots, we compare the two baselines on their 10000 unselected samples with the largest DoCL objective values, which are the better-predicted samples as implied by Figure 11.2 and the objective formulation in Eq. (4.12). The middle and bottom plots show that by selecting the highest-scored samples as in Baseline2, we can make greater learning progress and achieve better prediction accuracy on these better-predicted samples.

Therefore, the training dynamics observed on the three baseline curricula justify the proposed DoCL objective for dynamics optimization and motivate us to develop a curriculum learning method based on selecting samples with higher score $\hat{a}_t(i)$. We also provide similar empirical results on CIFAR100 in the following Figure 11.4.

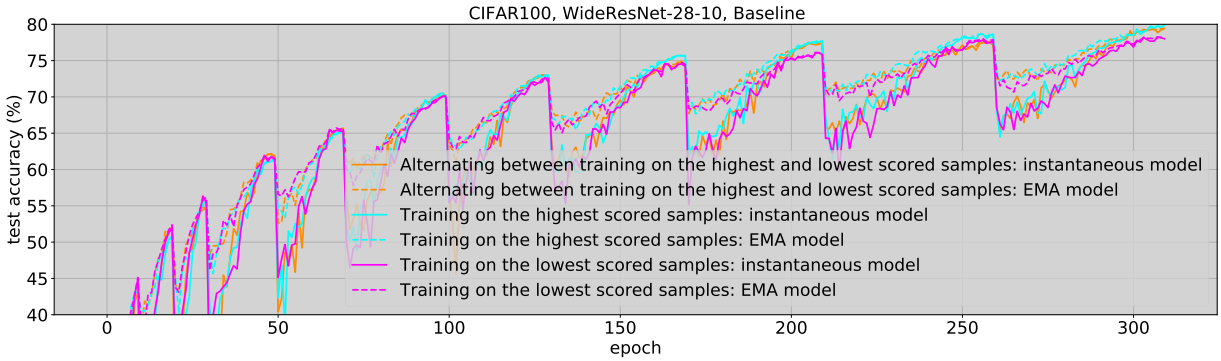

Figure 11.4: Test set accuracy of WideResNet-28-10 (instantaneous model) and its exponential moving average (EMA model) during the course of training when using the three data selection curricula.

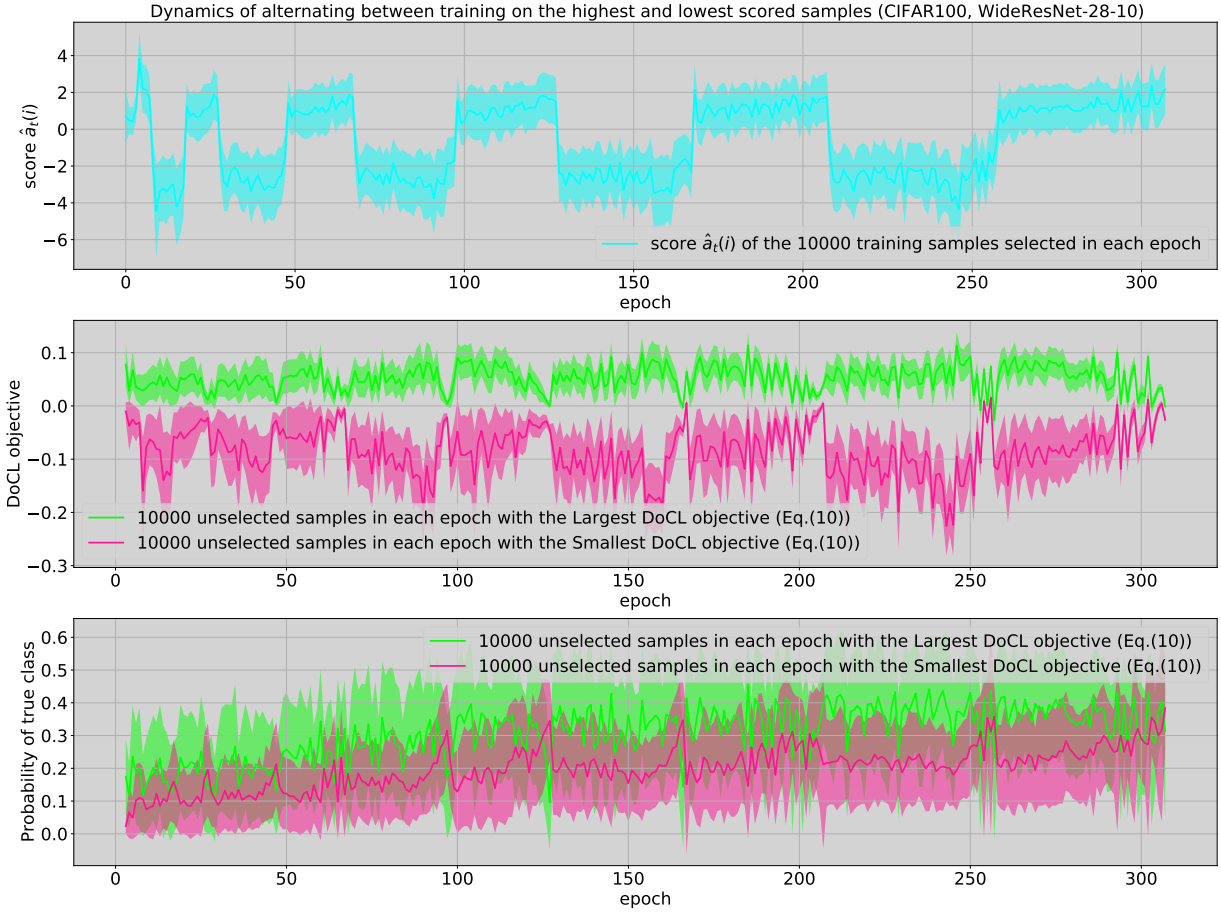

Figure 11.5: *Training alternates between the highest and lowest scored samples*: Dynamics (mean±std) for (**Top**) the score $\hat{a}(i)$ (Eq. (8.5)) of the selected samples, (**Middle**) the DoCL objective (Eq. (4.12)) values of unselected samples, and (**Bottom**) the output true-class probability for unselected samples. We split the unselected samples in each epoch into two groups with the largest/smallest DoCL objective values.

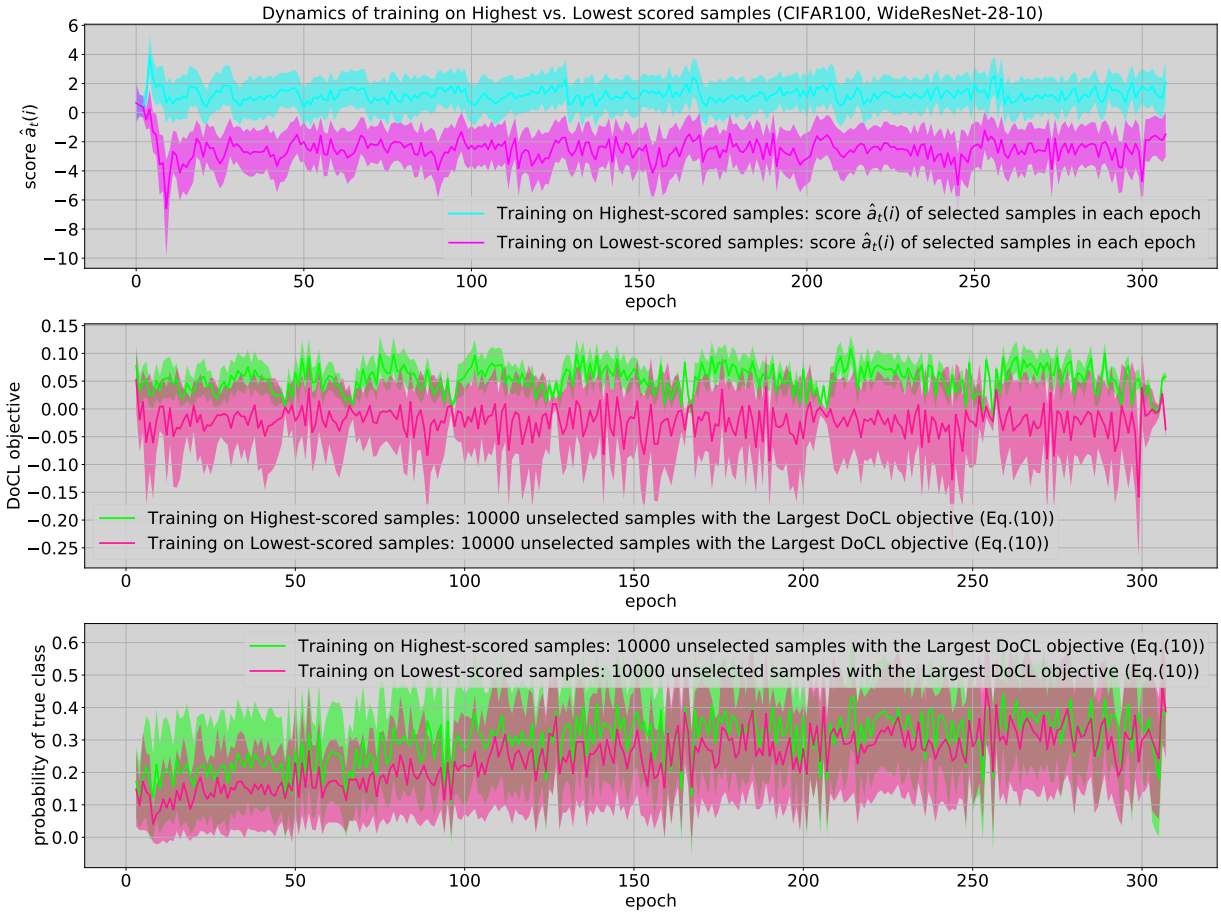

Figure 11.6: *Training with highest-scored vs. lowest-scored samples*: Dynamics (mean±std) for (**Top**) the score $\hat{a}(i)$ (Eq. (8.5)) of the selected samples, (**Middle**) the DoCL objective (Eq. (4.12)) values and (**Bottom**) the output true-class probability for unselected samples with the largest DoCL objective values.

11.2 Dynamics-optimized Curriculum Learning (DoCL) Algorithm

Algorithm 6 Dynamics-optimized Curriculum Learning (DoCL)

1: **input:** $\{(x_i, y_i)\}_{i=1}^n$, $\ell(\cdot,\cdot)$, $f(\cdot;\theta)$,
$\{\eta_t\}_{t=0}^{T_\kappa}$, $\{T_i\}_{i=0}^{\kappa}$, $\gamma_k \in [0,1]$, $k_{\min}$
2: **initialize:** $T_{-1} = 0$, $k = n$, $\rho_i = 0, g_i = f(x_i)$
3: **for** $j \in \{0, \cdots, \kappa\}$ **do**
4: **for** $t \in \{T_{j-1}, \cdots, T_j\}$ **do**
5: **if** $t < T_0$ or $t = T_j$ **then**
6: Uniform sampling $S_t \subseteq [n]$ up to size n;
7: Update θ by large-batch SGD with learning rate η_t to minimize L2 loss on S_t;
8: **else**
9: $S_t \leftarrow$ Draw k samples with probability $\propto \hat{a}_t(i)$;
10: *Optional*: prune S_t to a diverse subset by submodular maximization in Eq. (11.1);
11: Update θ by mini-batch SGD with learning rate η_t to minimize the task's loss $\ell(\cdot)$ on S_t;
12: **end if**
13: **for** $i \in S_t$ **do**
14: Estimate linear dynamics of $f(x_i)$:
$\rho_i \leftarrow \rho_i + \eta_t, \frac{\partial f(x_i)}{\partial t} = \frac{f(x_i) - g_i}{\rho_i}$;
15: Restore $\rho_i \leftarrow 0$ and $g_i \leftarrow f(x_i)$;
16: Compute $a_t(i)$ by Eq. (8.4) (regression) or Eq. (8.7) (classification);
17: Update $\hat{a}_{t+1}(i)$ using Eq. (8.5);
18: **end for**
19: **end for**
20: Reduce training set size: $k \leftarrow \max\{k_{\min}, \gamma_k \times k\}$;
21: **end for**

In this section, we will develop a new practical curriculum learning algorithm based mainly on the above dynamics-optimization strategy. It also integrates other techniques to make it more efficient and compatible with current deep learning schemes. We provide its detailed procedures in Algorithm 6 and subsequently elaborate on its major steps in the following.

Warm starting. To initialize the scores, at the beginning we run T_0 epochs of large-batch SGD (line 5-7) to minimize the L2 loss on the whole training set. These warm-start epochs provide accurate estimates of the scores in Eq. (8.4) (regression) or Eq. (8.7) (classification), in which the linear dynamics should be estimated under the full gradient flow (rather than stochastic gradient

flow) that minimizes the L2 loss on a training set D drawn from the data distribution $\mathcal{D}$.

Cyclical curriculum learning. We train the model for multiple (κ) episodes/cycles with an increasing number of steps (i.e., $T_{j+1} - T_j > T_j - T_{j-1}$ for $\{T_j\}_{j=1}^{\kappa}$ in Algorithm 6), where each episode starts with a large or rapidly increasing learning rate, which gradually decays towards zero by a predefined function (e.g., cosine or exponent). The learning rate decay results in a fast convergence to local minima, while its surge at the beginning of each episode helps to quickly jump out from the previous local minima. Hence, cyclical learning rates [192] such as the cosine annealing schedule [142] can quickly jump between different local minima on the loss landscape and explore more regions without being trapped in challenging local minima. It is a perfect match to our strategy since it leads to more exploration of the training dynamics under different learning rates, which improves the estimates of the scores. Moreover, at the end of each episode (line 20), we reduce the training set size k because more samples have their predictions converging to the ground truth as the training proceeds. In addition, we apply a large-batch training epoch (similar to the ones during warm starting) to update the scores (line 5-7).

Estimate the linear dynamics under varying learning rates. For computing $a_t(i)$ (line 16), we need to estimate the linear dynamics $\partial f(x)/\partial t$ in continuous time from the observations of $f(x)$ at discrete time steps. Since the learning rate η_t can change over time, and a larger learning rate leads to greater changes in $f(x)$, we estimate $\partial f(x)/\partial t$ at step t by $(f_t(x) - f_{t'}(x))/\sum_{q=t'}^{t} \eta_q$, where t' is the last step before t when x is selected for training. In line 14-15 of Algorithm 6, we update ρ_i and g_i to keep a record of $\sum_{q=t'}^{t} \eta_q$ and $f_{t'}(x)$ for sample-i, which is used to estimate the linear dynamics.

Update the scores by using dynamics computed on S_t. Theoretically, the scores can only be updated during the warm start epochs at the beginning of the algorithm and the update epoch at the end of each episode. In other steps (line 9-11), since we instead apply mini-batch training on a possibly biased subset S_t (i.e., not guaranteed to be i.i.d. drawn from $\mathcal{D}$) and minimize a loss determined by the task (i.e., not always to be the L2 loss), the resulting training dynamics

$\partial f(x)/\partial t$ can be different from the one required in Eq. (8.4) and Eq. (8.7). However, in practice, by encouraging more exploration on samples with small $\hat{a}_t(i)$ when sampling S_t (line 9), we find that the byproducts of those training steps can also be leveraged to update $\hat{a}_t(i)$ (line 13-18) and produce compelling performance.

Weighted sampling. The problem formulations in Eq. (4.8) and Eq. (4.12) suggest directly selecting samples with the largest scores $\hat{a}_t(i)$. For better exploration, however, we instead apply a weighted sampling of S_t based on the scores (line 9). We can also trade off exploration vs. exploitation using strategies from online learning methods. For example, we can sample S_t from a Boltzmann distribution, i.e., $\Pr(i \in S_t) = \exp(\hat{a}_t(i)/\tau)$, where τ is a temperature parameter. We can additionally apply exponential weights similar to Exp3 [9] if we assume the feedback $a_t(i)$ is more adversarial than entirely stochastic. The momentum $a_t(i)$ can either increase or decrease during different training stages so it is not entirely stochastic. It is neither purely adversarial since SGD on a complicated loss landscape does not play against the curriculum. In this case, we can additionally re-scale $a_t(i) \leftarrow a_t(i)/\Pr(i \in S_t)$ after line 16. It encourages more exploration since x with a smaller probability is more likely to be selected in the future.

Further prune S_t to a diverse subset. We can further reduce training time in early stages when k is large by extracting a small and diverse/representative subset of S_t. Inspired by MCL introduced in Section 2.2 and Chapter 9, at line 10, we reduce S_t to a subset of size $k'_t = \gamma_{k'} k_t$ ($0 < \gamma_{k'} \leq 1$) by (approximately) solving the following submodular maximization problem:

$$\max_{S \subseteq S_t, |S| \leq k'} \sum_{i \in S} \hat{a}_t(i) + \lambda_t F(S), \tag{11.1}$$

where $F : 2^{S_t} \rightarrow \mathbb{R}_+$ is a submodular function [66] so we can exploit fast greedy algorithms [158, 152, 153] to solve Eq. (11.1) with an approximation guarantee. We gradually reduce preference for diversity as training proceeds via reducing λ_t by a factor $0 \leq \gamma_\lambda \leq 1$ at each step.

In Section 15.3, on nine different datasets, DoCL significantly outperforms random mini-batch SGD and recent curriculum learning methods both in terms of efficiency and final performance.

Chapter 12

TIME-CONSISTENCY CURRICULUM FOR SEMI-SUPERVISED LEARNING

In this chapter, we propose the "time-consistent semi-supervised learning (TC-SSL)" algorithm based on the time-consistency (TC) score introduced in Section 8.3, which measures the correctness of a pseudo label produced by a model by its consistency over time. In the following, we will first introduce the training objective of TC-SSL at each step and explain the importance of selecting time-consistent samples to the training stability when using this objective. We then present the TC-SSL algorithm, which selects samples by their TC according to a curriculum that selects more unlabeled samples as training progresses. We then discuss some practical modifications to TC-SSL that bring further improvements.

12.1 Training Objective for Semi-Supervised Learning

We use two types of self-supervised losses which cooperate with each other to encourage the output consistency between similar samples on the data manifold. In particular, given a sample x and one of its augmentation $G(x)$, we minimize the consistency loss defined as the difference between $f(x)$ and $f(G(x))$, i.e., the neural net outputs on these two samples. In order to obtain a pseudo target robust to unbounded noise over training steps, we follow the mean teacher [209] method and instead use an exponential moving average of the neural net in-training to generate the pseudo target, i.e.,

$$\overline{f^t}(x) \triangleq f(x; \overline{\theta^t}), \overline{\theta^t} \triangleq \gamma_\theta \theta^{t-1} + (1 - \gamma_\theta)\overline{\theta^{t-1}}, \tag{12.1}$$

where θ^t is the neural net parameters at step t and $\overline{\theta^t}$ is recursively defined as the moving average of θ^t over time with discount factor $\gamma_\theta \in [0, 1]$. Although $\overline{\theta^t}$ tends to produce a smooth target over time, it cannot filter out time-inconsistent samples whose outputs change dramatically over time and their smoothed targets still suffer from large variance and low entropy. The consistency loss on x is defined as

$$\ell_{cs}^t(x;\theta) \triangleq \|f(G(x);\theta) - \overline{f^t}(x)\|_p, \tag{12.2}$$

where we set $p = 2$ in our experiments and the above loss aims to minimize the ℓ_2 distance between the two outputs. In order to stabilize back-propagation on $\ell_{cs}^t(x;\theta)$, $\overline{f^t}(x)$ is often treated as a constant pseudo target [209] (i.e., no back-propagation through $\overline{f^t}(x)$ to x) of $f(G(x))$. Hence, we only minimize the loss w.r.t. the current θ^t. In practice, we can further consider replacing the pseudo target $\overline{f^t}(x)$ with an average over different augmentations and multiple time steps, e.g., $1/hm \sum_{t=T-h}^{T} \sum_{i=1}^{m} \overline{f^t}(G_i(x))$, which might be more accurate [21]. For simplicity and efficiency, we do not adopt this form and use $\overline{f^t}(x)$ in our experiments.

Contrastive loss [38, 216, 37] enforces the distance between augmentations of the same sample to be smaller than distances between other samples (or other samples' augmentations). Specifically, we use the NT-Xent loss [37] equipped with the mean teacher prediction $\overline{f^t}(\cdot)$ for the positives and negatives. Given a dictionary D of "other" data and/or their augmentations (negatives) to compare with, the loss is defined as follows: , the loss is

$$\ell_{ct}^t(x;\theta) \triangleq -\log \frac{\exp(\text{cossim}[f(G(x);\theta), \overline{f^t}(G'(x))])}{\sum_{z \in \{G'(x)\} \cup D} \exp(\text{cossim}[f(G(x);\theta), \overline{f^t}(z)])}, \tag{12.3}$$

where $G(x)$ and $G'(x)$ can be either two different augmentations of x or two distinct instantiations of the same random augmentation policy applied to x, τ is a temperature parameter, and $\text{cossim}[x, z] \triangleq \frac{\langle x,z \rangle}{\|x\|_2 \cdot \|z\|_2}$ denotes the cosine similarity. In the denominator, both the positives in $G'(x)$ and the

negatives in D are included. By choosing an appropriate τ*, NT-Xent loss outperforms other contrastive losses such as InfoNCE [216], as shown in Section 5.1 of [37]. There may exist other types of contrastive losses that outperform the two and we use NT-Xent here as an example. It is worth noting that our method is not limited and can be directly applied to other types of contrastive loss.

The contrastive loss above can be explained as a cross entropy loss, where the predicted probability distribution (computed by applying softmax to the $|D|+1$ similarities in the denominator) is defined over all the $|D|+1$ samples that measures the probability of each sample being similar to $G(x)$, and the pseudo label is 1 for the same sample's augmentation $G'(x)$ and 0 for other samples. Similar to consistency loss, all the quantities with $\overline{f^t}(\cdot)$ in Eq. (12.3) are treated as constants during back-propagation. Recent work [80] shows that minimizing the contrastive loss requires a sufficiently large dictionary size $|D|$ to achieve improvements. In our experiments, $|D|$ is often set ≥ 1000. During training, we keep a fixed-length queue of samples' outputs in memory as the dictionary D and replace the oldest outputs in the queue by the latest training batch's outputs.

These two types of self-supervised losses constitute only one part of our SSL optimization objective. The consistency and contrastive losses provide two complementary self-supervision strategies, one maximizing the absolute similarity between similar samples, while the other enforcing that the relative similarity between similar samples should be much larger than the one between distant samples. When applied together with data augmentation (which can cover a larger local region around each sample on the data manifold) and expressive neural nets (which encodes the prior of data structure and enjoys universal approximation capability), they can effectively learn a locally consistent and smooth model over the data manifold, without requiring ground truth labels for most samples. Nor does the method use similarity between all the samples (as most earlier works do), also such graph-based methods could be incorporated in our work possibly further improving

*In experiments, the best τ normally lies in $[0.001, 0.1]$ because we compute the exponential of a bounded cosine similarity between $[-1, 1]$.

performance. To the best of our knowledge, TC-SSL is the first attempt to combine consistent and contrastive losses together for SSL.

The two self-supervised losses enforce local consistency but not time-consistency. For a sample or its augmentation, its pseudo targets in the loss (i.e., all the terms containing $\overline{f^t}(\cdot)$ in Eq. (12.2) and Eq. (12.3)) at different training steps us time-variant and can be inconsistent, since the neural net outputs on the sample or its nearby region may change drastically during training. There are several possible reasons for such instability on individual samples: a non-smooth activation function, a complicated neural net structure, a steep learning rate schedule, or too complicated a data augmentation strategy. When the training targets significantly change over time, the learning could be very inefficient or even diverge and suffer from concept drift. Therefore, selecting time-consistent samples for more consistent self-supervision is essential to the widely used strategy of combining data augmentation with self-supervised losses.

For a labeled sample (x, y) with ground truth class y, we minimize their cross entropy loss, i.e.,

$$\ell_{ce}(x, y; \theta) \triangleq -\log \frac{\exp(f(G(x); \theta)[y])}{\sum_{y'} \exp(f(G(x); \theta)[y'])}, \tag{12.4}$$

In order to minimize the entropy of the outputs for unlabeled data [70, 122], we also minimize a cross entropy loss with $y^{t-1}(x)$ as the pseudo label of each selected unlabeled data x, i.e.,

$$\ell^t_{ce}(x; \theta) \triangleq -\log \frac{\exp(f(G(x); \theta)[y^{t-1}(x)])}{\sum_{y'} \exp(f(G(x); \theta)[y'])}, \tag{12.5}$$

The overall training objective of TC-SSL at step t is

$$L^t(\theta) = \frac{1}{|\mathscr{L}|} \times \sum_{(x,y)\in\mathscr{L}} \ell_{ce}(x, y; \theta) + \frac{1}{|S^t|} \times \sum_{x\in S^t} \left[\lambda_{cs}\ell^t_{cs}(x; \theta) + \lambda_{ct}\ell^t_{ct}(x; \theta) + \lambda_{ce}\ell^t_{ce}(x; \theta)\right], \tag{12.6}$$

where $\mathscr{L}$ denotes the set of all labeled data and $S^t \subseteq \mathscr{U}$ is the subset of unlabeled data selected by TC at step t from the unlabeled data set $\mathscr{U}$, and λ_{cs}, λ_{ct}, and λ_{ce} are weights for different types of loss that can be tuned on a validation set.

12.2 Time-Consistent Semi-Supervised Learning (TC-SSL) Algorithm

We provide the complete description of TC-SSL in Algorithm 7. At step t, TC-SSL aims to select a subset of unlabeled data S^t with higher TC score $c^t(x)$ and then minimizes the objective in Eq. (12.6) w.r.t. the neural net weights θ^t by taking gradient steps. We can either select the top-k unlabeled samples with the largest $c^t(x)$ as S^t (line-8) or apply a weighted sampling of S^t according to $\Pr(x \in S^t) \propto \exp(c^t(x))$ (line-9). The optimization of $L^t(\theta^t)$ can be one or several steps of gradient descent or SGD (line-10). We denote the update of θ^t produced by the adopted optimizer as $\pi(\nabla_\theta L^t(\theta^t), \eta^t)$, where η^t contains all hyperparameters of the optimizer at step t, e.g., the learning rate. We keep updating TC score $c^t(x)$ for every unlabeled data $x \in \mathscr{U}$ (line-18) as a (negative) moving average of $a^t(x)$ (line-16), and update a moving average of θ^t over time (line-19) as the mean teacher [209] providing pseudo targets for the two self-supervised losses in $L^t(\theta^t)$.

Since the TC score $c^t(x)$ is an accumulated statistic over multiple time steps, we need a sufficiently long horizon to estimate it accurately and stably before using. Moreover, the change of neural net outputs on a sample in a single step can be easily perturbed by SGD randomness, the learning rate change, and neural net weight changes during early training. Therefore, we apply T_0 warm starting epochs (line-6) that train the neural net using only the labeled data before selecting any unlabeled data for self-supervision. In addition, there may be fewer time-consistent unlabeled samples in earlier stages without sufficient training. As more training occurs with consistent targets, the neural nets' accuracy and confidence improve, and we expect an increase in the number of time-consistent samples. Hence, over the course of training, we gradually increase the number of unlabeled samples selected into S^t (line-20). This yields a curriculum for SSL that in early stages

Algorithm 7 Time-Consistent SSL (TC-SSL)

input: $\mathcal{U}, \mathcal{L}, \pi(\cdot;\eta), \eta^{1:T}, f(\cdot;\theta), G(\cdot)$;
hyperparameters: $T_0, T, \lambda_{cs}, \lambda_{ct}, \lambda_{ce}, \gamma_\theta, \gamma_c, \gamma_k$;
initialize: θ^0, k^1;
for $t \in \{1, \cdots, T\}$ **do**
  if $t \leq T_0$ **then**
    $\theta^t \leftarrow \theta^{t-1} + \pi\left(\sum_{(x,y)\in\mathcal{L}} \nabla_\theta \ell_{ce}(x, y; \theta^{t-1}); \eta^t\right)$
  else
    $S^t = \text{argmax}_{S:S\subseteq\mathcal{U},|S|=k^t} \sum_{x\in S} c^t(x)$ or
    Draw k^t samples from $\Pr(x \in S^t) \propto \exp(c^t(x))$;
    $\theta^t \leftarrow \theta^{t-1} + \pi\left(\nabla_\theta L^t(\theta^{t-1}); \eta^t\right)$ (ref. Eq. (12.6));
  end if
  $p^t(x) \leftarrow \frac{\exp(f(x;\theta^t)[y])}{\sum_{y'=1}^{C} \exp(f(x;\theta^t)[y'])}$, $\forall y \in [C], x \in \mathcal{U}$;
  if $t = 1$ **then**
    $\overline{\theta^t} \leftarrow \theta^t, c^t(x) \leftarrow 0, \ \forall x \in \mathcal{U}$
  else
    Compute $a^t(x)$ (ref. Eq (8.10)), $\forall x \in \mathcal{U}$;
  end if
  $c^{t+1}(x) \leftarrow \gamma_c(-a^t(x)) + (1-\gamma_c)c^{t-1}(x), \ \forall x \in \mathcal{U}$;
  $\overline{\theta^{t+1}} \leftarrow \gamma_\theta \theta^t + (1-\gamma_\theta)\overline{\theta^t}$;
  $k^{t+1} \leftarrow (1+\gamma_k) \times k^t$;
end for

effective saves computation and avoids perturbation caused by inconsistent unlabeled data, and in later stages fully explores the information brought by the reserve of unlabeled data.

In MixMatch [22] and MixUp [243], augmentation over unlabeled and labeled data can significantly improve SSL performance. One possible reason is that minimizing the self-supervised loss on linear interpolations between two samples enforces the local consistency and smoothness over larger regions of the data manifold. In TC-SSL, after single-sample augmentation (if any), we apply standard MixUp between all training samples in $\mathcal{U} \cup \mathcal{L}$, but only use the resulted MixUp samples in cross entropy loss, i.e., Eq. (12.4) and Eq. (12.5). Unlike MixMatch, we do not need to modify MixUp and simply treat each unlabeled sample as labeled with a pseudo class when mixing its target with another (labeled or unlabeled) sample. For the two self-supervised losses, we still use

the single-sample augmentations before MixUp because: (1) the consistency loss already uses a soft pseudo label but MixUp further reduces its entropy; and (2) the contrastive loss aims to discriminate different samples but replacing them with their interpolations will weaken the effect.

12.3 Practical and Heuristic Improvements

The first two improvements are specifically designed for TC-SSL, while the remaining three improvements follow the previous methods for semi-supervised learning.

Calibrate the time-consistency by the learning rate: Since the learning rate η^t can change over time, and large learning rates lead to greater changes on model outputs, the scales of $a_t(x)$ might change accordingly. Hence, we calibrate $a_t(x)$ (via $a^t(x) \leftarrow a^t(x)/\eta^t$) before using it.

Exponential weight for exploration-exploitation trade-off: During early training, the weighted sampling in line-9 of Algorithm 7 usually works better than selecting the top-k^t samples (line-8) since the randomness avoids selecting the same samples repeatedly and encourages exploration of new unlabeled data, whose TC might be improved once being trained. In practice, we can achieve a better trade-off between exploration and exploitation by using similar techniques to Exp3 [9], i.e., instead of applying line-8 or line-9, we sample S^t according to a smoothed probability $\Pr(x \in S^t) \propto \exp(\sqrt{2\log|\mathcal{U}|/|\mathcal{U}|} \times c^t(x))$ and then re-scale $a^t(x)$ for all selected $x \in S^t$ by their probability of being selected, i.e., $a^t(x) \leftarrow a^t(x)/\Pr(x \in S^t)$. Thereby, x with smaller probability will get more of a chance to be selected in the future.

Decrease the entropy of pseudo target: Following [70, 155, 22, 122], we also decrease the entropy of pseudo target distribution in the contrastive loss by multiplying a temperature parameter $\nu > 1$ [†] to the quantities inside the exponential in Eq. (12.3).

[†] e.g., we use $\nu = 10$ in all experiments.

Prune S^t using the confidence statistics on a validation set: Although TC is a better metric than confidence for unlabeled data selection, confidence is still not useless. For instance, it is not efficient to train models with unlabeled samples of either extremely high or low confidence since the former contributes very small gradients while the latter's pseudo target suffers from high entropy. Therefore, we remove them from S^t by applying two thresholds computed on a validation set, which are the $b\%$-percentile and $(100-b)\%$-percentile of the confidence computed on the validation set [‡]. We do not use the statistics on training set $\mathcal{U} \cup \mathcal{L}$ since the model can be over-confident on samples being trained.

Duplicate labeled samples: When applying TC-SSL with MixUp, we duplicate the labeled samples so that an unlabeled sample has higher a chance of being mixed with a labeled sample. This improves training robustness since mixing the incorrect pseudo target of an unlabeled sample with the correct label of a labeled sample is less harmful. However, unlike MixMatch, we do not duplicate the labeled samples to the same amount of unlabeled samples. Instead, we start using fewer duplicates [§] than MixMatch — this amount is comparable to the amount of unlabeled data selected in the first iteration, and we gradually reduce the duplicates as training proceeds. Thanks to the TC score, the selected samples are likely to be correct with stable pseudo targets and we do not need to spend much compute on the labeled data to keep training robust and stable.

In Section 16.1, we provide a detailed and thorough evaluation of TC-SSL algorithm on multiple semi-supervised learning benchmark datasets and compare it with several SoTA semi-supervised learning approaches.

[‡]e.g., we use $b = 10$ in experiments

[§]e.g., we use $k^1 = |S^1|$, which is $0.1|\mathcal{U}|$ in all experiments.

Chapter 13

ROBUST CURRICULUM LEARNING (ROCL)

In this chapter, we propose "Robust Curriculum Learning (RoCL)" algorithm for noisy-label learning (NLL). RoCL is built upon the score functions introduced in Section 8.4, the problem formulation with tilted loss, and the scheduling strategy studied in Chapter 3. The score functions are designed for both the clean label detection and pseudo-label selection tasks for NLL. By adjusting the two temperature parameters τ_1 and τ_2 in Eq. (3.7) and the trade-off weight λ in Eq. (3.6), RoCL can smoothly interpolate between the two selection tasks and control their trade-off. We formally introduce the RoCL algorithm and its detailed procedures in the following.

In particular, RoCL smoothly transitions between two phases: (1) detection and supervised training on clean data; and (2) relabeling and self-supervision on noisy data. In RoCL, we train the model for multiple episodes, each starting from phase(1) and gradually moving to phase(2). Unlike existing approaches, we only select samples with accurate given/pseudo labels that are most informative to the current model training. The data selection criterion in each phase is based on the corresponding score developed in Section 8.4, which captures the dynamics of per-sample loss or output consistency (across multiple data augmentations). Similar to DIH score in Section 8.1 and TC score in Section 8.3, those scores overcome the instability of instantaneous feedback and does not incur any additional inference cost.

In addition, by adjusting a temperature parameter, the criterion can interpolate between the two phases and keep the training focusing on the data that the model mostly needs to improve on, e.g., clean data with unsatisfying output consistency or wrongly-labeled data with accurate pseudo labels. Thus, we can fully exploit both clean and noisy data more efficiently with less risk of introducing

extra noise or error accumulation. We further show that our data selection can be derived from a novel optimization formulation for robust curriculum learning. We evaluate our method on multiple noisy learning benchmarks and show that our method outperforms a diverse set of recent noisy-label learning approaches.

13.1 Robust Curriculum Learning (RoCL) Algorithm

We describe our curriculum learning algorithm RoCL in Algorithm 8. We denote the update of θ_t produced by the adopted optimizer as $h(\nabla_\theta F(\theta_t), \eta)$, where η contains all hyperparameters of the optimizer at step t, e.g., the learning rate. We denote $\ell_t(i)$ as a shorthand notation for $\ell(f(x_i; \theta_{t-1}), y_i)$.

Algorithm 8 Robust Curriculum Learning (RoCL)

1: **input:** $\{(x_i, y_i)\}_{i=1}^n$, $h(\cdot;\eta)$, $\ell(\cdot,\cdot)$, $f(\cdot;\theta)$, $T_{0:K}$; $\tau_1 < 0, \tau_T > 0$; $\lambda_1, \lambda_T \in [0,1]$; $\gamma, \gamma_b \in [0,1]$
2: **initialize:** θ_0, $b_0 \in (0, n)$, $l_0(i) = c_0(i) = 0\ \forall i \in [n]$
3: **for** $k \in \{0, \cdots, K\}$ **do**
4: Schedule $\tau_{1:T_k}$ and $\lambda_{1:T_k}$ by Eq. (13.1)-(13.2);
5: **for** $t' \in \{1, \cdots, T_k\}$ **do**
6: $t \leftarrow t' + T_{k-1}$;
7: **if** $k = 0$ **then**
8: $S_t \leftarrow [n]$;
9: $\theta_t \leftarrow \theta_{t-1} + h\left(\nabla_\theta \frac{1}{n} \sum_{i=0}^n \ell_t(i); \eta\right)$;
10: **else**
11: Draw a subset $S_t \subseteq [n]$ of b_k samples according to probability $\mathscr{P}_t$ in Eq. (3.6) with $\tau_1 = \tau_{t'}, \tau_2 = \tau_{T_k - t'}, \lambda = \lambda_{t'}$;
12: $\theta_t \leftarrow \theta_{t-1} + h\left(\frac{1}{b_k} \sum_{i \in S_t} G_t(i); \eta\right)$ (Eq. (3.4));
13: **end if**
14: Update $l_{t+1}(i)$ and $c_{t+1}(i)$ by Eq. (8.24) and Eq. (8.26);
15: **end for**
16: $b_{k+1} \leftarrow (1 + \gamma_b) \times b_k$;
17: **end for**

We apply a warm starting episode of a few epochs (e.g., Line 5-10) over all the data and given labels with label smoothing to obtain stable EMA metrics. After that, we apply multiple episodes

of curriculum learning, each including a sequence of steps following the curriculum at the end of Section 3.3 for data selection per step (Line 6-14). We repeat the transition between clean data learning to noisy data learning for K episodes to avoid getting trapped in a local minimum dominated by a small set of clean/noisy data or a specific type of loss. It also reinforces the memorization of clean labels and correct pseudo labels learned in previous episodes. Moreover, under the coupling strategy of τ_1 and τ_2, each episode is encouraged to explore the clean/noisy data that the previous episode fails to learn. Considering the undertrained model (producing inaccurate pseudo labels) and the relatively high variance of the EMA scores at the earlier episodes, we start from a small budget for the selected subset size and gradually increase in later episodes.

13.2 Compound Scheduling in RoCL Algorithm

RoCL applies a scheduling strategy of $(\tau_1, \tau_2, \lambda)$ introduced in Section 3.3 to achieve a smooth transition from supervised learning to self-supervised learning and to keep the training data selected for each phase the most informative. In practice, we can further reduce the hyperparameters: (1) given the sequence for τ_1 in the curriculum as $\tau_{1:T}$, we can reverse it as the sequence for τ_2, i.e., $\tau_{T:1}$; (2) we can make λ monotone increase with τ_1, e.g., setting the initial value $\lambda_1 = a\tau_1 + b$ and ending value $\lambda_T = a\tau_T + b$, solving this linear system for a and b, which generate $\lambda_{1:T} = a\tau_{1:T} + b$. To generate the whole schedule of $\tau_{1:T}$ in each episode, we can apply any monotone interpolation between τ_1 and τ_T whose values are predefined. Let $g : \mathcal{R} \mapsto [-\sigma, \sigma]$ be an invertible monotone continuous function. We define the interpolation between τ_1 and τ_T as follows, $\forall t \in [T]$,

$$\tau_t = \frac{\tau_T - \tau_m}{\sigma} \times \left[g(\sigma_t) - \frac{g(-\sigma) + g(\sigma)}{2} \right] + \tau_m,\ \sigma_t = g^{-1}(-\sigma) + \frac{2t}{T} g^{-1}(\sigma),\ \tau_m = \frac{\tau_1 + \tau_T}{2}. \tag{13.1}$$

Note the $g(\sigma_t)$ produces interpolation values between $[-\sigma, \sigma]$ that correspond to T evenly spaced input $\sigma_t \in [g^{-1}(-\sigma), g^{-1}(\sigma)]$. In our curriculum for each episode, we need to keep a high quality of

the selected clean (pseudo) labels in earlier (later) stages and make the exploration stages in between shorter since their selected labels contain more noise. Therefore, we choose "s"-shaped functions such as $\tanh$ or the logistic function for the interpolation. In the experiments of Section 16.2, we use $g(\cdot) = \tanh(\cdot)$ and pick $\sigma = 0.95$. We illustrate Eq. (13.1) and visualize our choice of $g(\cdot)$ and the resulting τ_t in Figure 13.1.

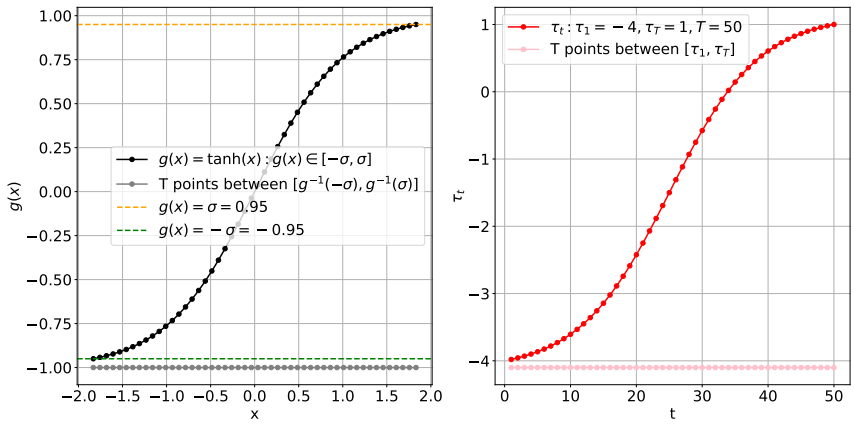

Figure 13.1: Illustration of Eq. (13.1) and visualization of our choice for $g(\cdot)$ and the resulted τ_t when $T = 50$. We use $g(\cdot) = \tanh(\cdot)$ (which can be other "S"-shape functions) and $\sigma = 0.95$ in our experiments. Here, we map the points on the black curve in the left plot to the points on the red curve in the right plot. Each gray point on the bottom of the left plot is from the T evenly spaced x-coordinates between the x-interval $[g^{-1}(-\sigma), g^{-1}(\sigma)]$. We scale them to the T t-coordinates in the bottom of the right plot (i.e., $t = 1, 2, \cdots, 50$), which associates with T τ_t values represented by the red points between $[\tau_1, \tau_T]$ on the red curve.

The corresponding schedule for λ can then be defined as an affine transformation of $\tau_{1:T}$:

$$\forall t \in [T], \lambda_t = a_\lambda(\tau_t - \tau_1) + \lambda_1, a_\lambda = \frac{\lambda_T - \lambda_1}{\tau_T - \tau_1}. \tag{13.2}$$

In the experiments of Section 16.2, the above scheduling method effectively reduces the hyparameters needed to be tuned for the scheduling of $(\tau_1, \tau_2, \lambda)$. In addition, RoCL adopting this scheduling strategy performs promisingly and stably on several challenging NLL benchmarks.

Chapter 14

DIVERSE ENSEMBLE EVOLUTION

In this chapter, we firstly propose an efficient continuous-combinatorial optimization algorithm to address the diverse ensemble learning problem introduced in Eq. (5.3) of Chapter 5. We analyze several interesting theoretical properties of this algorithm. We then develop a curriculum learning algorithm for "Diverse Ensemble Evolution (DivE^2)", which solve a sequence of the optimization problem defined by a scheduling of its hyperparameters in its constraints and objective. In particular, the scheduling leads to a curriculum of a smooth transition from "model selecting data" to "data selecting model" along with a decreasing weight for the diversity terms in Eq. (5.3). While the first phase aims at developing diverse expertise of each model in the ensemble, the second phase enforces the expertise of all the models to be complementary so each training sample has to be covered by some of the models from the ensemble.

14.1 Ensemble Optimization with Data-Model Marriage

Eq. (5.3) is a hybrid optimization involving both a continuous variable W and a discrete variable A. It degrades to maximization of a piece-wise continuous function $H(W) \triangleq \max_{A\subseteq E, A\in\mathcal{I}_v\cap\mathcal{I}_u} G(A, W)$, with each piece defined by a fixed A achieving the maximum of $G(A, W)$ in a local region of W. Suppose that A is fixed, then maximizing $G(A, W)$ (or $H(W)$) consists of to m independent continuous minimization problems, i.e., $\min_{w_i} \sum_{v_j\in V(A\cap\delta(u_i))} \ell(v_j; w_i),\ \forall i \in [m]$. Here $V(A) \subseteq V$ denotes the samples incident to the set of edges $A \subseteq E$, so $V(A \cap \delta(u_i))$ is the subset of samples assigned to model u_i. When loss $\ell(\cdot; w_i)$ is convex w.r.t. w_i for every i, a global optimal solution to the above continuous minimization can be obtained by various off-the-shelf algorithms. When

Algorithm 9 SELECTLEARN$(k, p, \lambda, \gamma, \{w_i^0\}_{i=1}^m)$

Input: $\{v_j\}_{j=1}^n, \{l(\cdot; w_i)\}_{i=1}^m, \pi(\cdot; \eta)$
Output: $\{w_i\}_{i=1}^m$
Initialize: $w_i \leftarrow w_i^0 \ \ \forall i \in [m], t = 0$
while *not "converged"* **do**
  $W \leftarrow \{w_i^t\}_{i=1}^m$, define $G(\cdot, W)$ by W;
  $\hat{A} \leftarrow$ SUBMODULARMAX$(G(\cdot, W), k, p)$;
  if $G(\hat{A}, W) > G(A, W)$ **then**
    $A \leftarrow \hat{A}$;
  end if
  for $i \in \{1, \cdots, m\}$ **do**
    $-\nabla \hat{H}(w_i^t) \leftarrow \frac{\partial}{\partial w_i^t} \sum_{v_j \in V(A \cap \delta(u_i))} \ell(v_j; w_i^t)$;
    $w_i^{t+1} \leftarrow w_i^t + \pi\left(\{w_i^\tau, -\nabla \hat{H}(w_i^\tau)\}_{\tau \in [1,t]}; \eta^t\right)$;
  end for
  $t \leftarrow t + 1$;
end while

$\ell(\cdot; w_i)$ is non-convex, e.g., each model is a deep neural networks, there also exist many practical and provable algorithms that can achieve a local optimal solution, say, by backpropagation.

Suppose we fix W, then maximizing $G(A, W)$ reduces to the data assignment problem (a generalized bipartite matching problem [135], see Appendix [3] Sec. 5.4.3 for more details), and the optimal A defines one piece of $H(W)$ in the vicinity of W. Finding the optimal assignment is NP-hard since $G(\cdot, W)$ is a submodular function (a weighted sum of a modular and two submodular functions) and we wish to maximize over a feasibility constraint consisting of the intersection of two partition matroids ($\mathcal{I}_v \cap \mathcal{I}_u$). Thanks to submodularity, fast approximate algorithms [158, 152, 153] exist that find a good quality approximate optimal solution. Let $\hat{H}(W)$ denote the piecewise continuous function achieved when the discrete problem is solved approximately using submodular optimization, then we have $\hat{H}(W) \geq \alpha \cdot H(W)$ for every W, where $\alpha \in [0, 1]$ is the approximation factor.

Therefore, solving the max-max problem in Eq. (5.3) requires interaction between a combinato-

rial (submodular in specific) optimizer and a continuous (convex or non-convex) optimizer $\pi(\cdot;\eta)$ [*]. We alternate between the two optimizations while keeping the objective $G(A, W)$ non-decreasing. Intuitively, we utilize the discrete optimization to select a better piece of $\hat{H}(W)$, and then apply the continuous optimization to find a better solution on that piece.

Details are given in Algorithm 9. For each iteration, we compute an approximate solution $\hat{A} \subseteq E$ using submodular maximization SUBMODULARMAX (line 6); in lines 7-9 we compare $\hat{A}$ with the old A on $G(\cdot, W)$ and choose the better one; lines 10-13 run an optimizer $\pi(\cdot;\eta)$ to update each model w_i according to its assigned data. Algorithm 9 always generates a non-decreasing (assuming $\pi(\cdot;\eta)$ does the same using, say, a line search) sequence of objective values. With a damped learning rate, only small adjustments get applied to W and $G(\cdot, W)$. Thus, after a certain point the combinatorial part repeatedly selects the same A (and line 7 eventually is always false), so the algorithm then converges as the continuous optimizer converges. [†]

14.2 Theoretical Properties

An interesting viewpoint of the max-max problem in Eq. (5.3) is its analogy to K-means problems [140]. Eq. (5.3) strictly generalizes the kmeans objective, by setting $\gamma = \lambda = 0, k = 1$, p to be the number of desired clusters, and the loss to be the distance metric used in K-means (e.g., L2 distance), and the model to be a real valued vector of having the same dimension as x. Since K-means problem is NP-hard, our objective is also NP-hard.

We next analyze conditions for either of the constraints $(\mathcal{I}_v, \mathcal{I}_u)$ introduced in Section 5.4.1 to saturate. In the two extreme cases, we know that the "sample selecting model" constraint $\mathcal{I}_v$ saturates when $nk \ll mp$ (e.g., $k = 1$ and $p = n$), and the "model selecting sample" constraint $\mathcal{I}_u$

[*]The optimizer $\pi(\cdot;\eta)$ can be any gradient descent methods, e.g., SGD, momentum methods, Nesterov's accelerated gradient [160], Adagrad [55], Adam [107], etc. Here the first parameter $\cdot$ can include any historical solutions and gradients, and η is a learning rate schedule (i.e., learning rate is η^t for iteration t).

[†]Convergence is defined as the gradient $\nabla\hat{H}(W)$ w.r.t. W being zero. In practice, we use $\|\nabla\hat{H}(W)\| \leq \epsilon$ for a small ϵ.

saturates when $nk \gg mp$ (e.g., $k = m$ and $p = 1$). However, it is not clear what exactly happens between them. The following Lemma shows the precise saturation conditions of the two constraints, with proof details in Section 14.4.1.

Lemma 3. *If* SUBMODULARMAX *is greedy algorithm or its variant, the data assignment* $\hat{A}$ *produced by lines 6-9 in Algorithm 9 fulfills: (1)* $\mathcal{F}_v$ *saturates, i.e.,* $|\hat{A} \cap \delta(v)| = k, \forall v \in V$*, and* $|\hat{A}| = nk$*, if* $k < {}^{mp+p}/_{n+(p-1)}$*; (2)* $\mathcal{F}_u$ *saturates, i.e.,* $|\hat{A} \cap \delta(u)| = p, \forall u \in U$*, and* $|\hat{A}| = mp$*, if* $k > {}^{mp-p}/_{n-(p-1)}$*; (3) when* ${}^{mp+p}/_{n+(p-1)} \leq k \leq {}^{mp-p}/_{n-(p-1)}$*, we have* $|\hat{A}| \geq \min\{(k-1)+(m-k+1)p, (p-1)+(n-p+1)k\}$.

As stated above, we can think the objective $H(W)$ as a piecewise function, where each piece is associated with a solution to the discrete optimization problem. Since it is NP-hard to optimize the discrete problem, Algorithm 9 optimizes W on $\hat{H}(W)$, which is defined by the SUBMODULARMAX solutions, rather than on $H(W)$. Algorithm 9 has the following properties.

Proposition 3. *Algorithm 9: (1) generates a monotonically non-decreasing sequence of objective values* $G(A; W)$ *(assuming* $\pi(\cdot; \eta)$ *does the same) (2) converges to a stationary point on* $\hat{H}(W)$*; and (3) for any loss* $\ell(u, w)$ *that is* β*-strongly convex w.r.t.* w*, if* SUBMODULARMAX *has approximation factor* α*, it converges to a local optimum* $\hat{W} \in \operatorname{argmax}_{W \in \mathcal{K}} \hat{H}(W)$ *(i.e.,* $\hat{W}$ *is optimal in an local area* $\mathcal{K}$*) such that for any local optimum* $W^*_{loc} \in \mathcal{K}$ *on the true objective* $H(W)$*, we have*

$$\hat{H}(\hat{W}) \geq \alpha H(W^*_{loc}) + \frac{\beta}{2} \cdot \min\{(k-1)+(m-k+1)p, (p-1)+(n-p+1)k\} \cdot \|\hat{W} - W^*_{loc}\|_2^2. \quad (14.1)$$

The proof is in Section 14.4.2. The result in Eq. (14.1) implies that in any local area $\mathcal{K}$, if $\hat{W}$ is not close to W^*_{loc} (i.e., $\|\hat{W} - W^*_{\text{loc}}\|^2$ is large), the algorithm can still achieve an objective $\hat{H}(\hat{W})$ close to $H(W^*_{\text{loc}})$, which is a good approximate solution from the perspective of maximizing $G(A, W)$. Section 5.4.3 shows that the approximation factor is $\alpha = {}^1/_{2+\kappa_G}$ for the greedy algorithm, where κ_G is the curvature of $G(\cdot, W)$. When the weights λ and γ are small, κ_G decreases and

$G(\cdot, W)$ becomes more modular. Therefore, the approximation ratio α increases and the lower bound in Eq. (14.1) improves. For general non-convex losses and models (e.g., DNNs), Eq. (14.1) degenerates to a weaker bound: $\hat{H}(\hat{W}) \geq \alpha H(W^*_{\text{loc}})$.

14.3 Curriculum towards a Diverse Ensemble with Complementary Expertise

For a model ensemble to produce correct predictions, we require only that every sample be learnt by a few (small k) models. Optimizing Eq. (5.3) with small k from the beginning, however, might be harmful as the models are randomly initialized, and using the loss of such early stage models for the edge weights and small k could lead to arbitrary samples being associated and subsequently locked to models. We would, instead, rather have a larger k and more use of the diversity terms at the beginning. To address this, we design an ensemble curriculum to guide the training process and to gradually approach our ultimate goal.

Algorithm 10 Diverse Ensemble Evolution (DIVE^2)

Input: $\{(x_j, y_j)\}_{j=1}^n, \{w_i^0\}_{i=1}^m, \pi(\cdot;\eta), \mu, \Delta_k, \Delta_p, T$
Output: $\{w_i^t\}_{i=1}^m$
Initialize: $k \leq m, p \geq 1$ s.t. $mp \leq nk$,
$\lambda \in [0,1], \gamma \in [0,1]$
for $t \in \{1, \cdots, T\}$ **do**
  $\{w_i^t\}_{i=1}^m \leftarrow$ SELECTLEARN$(k, p, \lambda, \gamma, \{w_i^{t-1}\}_{i=1}^m)$;
  $\lambda \leftarrow (1-\mu)\cdot\lambda, \gamma \leftarrow (1-\mu)\cdot\gamma$;
  $k \leftarrow \max\{\lceil k - \Delta_k \rceil, 1\}, p \leftarrow \min\{\lfloor p + \Delta_p \rfloor, n\}$;
end for

In Section 5.4.1, we discussed two ($mp < nk$ and $mp > nk$) extreme training regimes. In the first regime, there are plenty of samples to go around but models may only choose a limited set of samples, so this encourages different models to improve on samples they are already good at. In the first regime, however, inter-model diversity is important to encourage models to become sufficiently different from each other. Intra-diversity is also important in the first regime, since it discourages models from being trained on entirely redundant data. In the second regime, there are plenty of models to go around but samples may choose only a limited number of models. Each model is then given a set of samples that it is particularly good at, and further training further specialization. Since all samples are assigned

models, this leads to complementary proficiencies covering the data space.

These observations suggest we start at the first regime $mp \leq nk$ with small p and large k, and gradually switch to the second regime with $mp \geq nk$ by slowly increasing p and decreasing k. In earlier stages, we also should set the diversity weights λ and γ to be large, and then slowly reduce them as we move towards the second regime. It is worth noting that besides intra-model diversity regularization, increasing p is also helpful to expand the expertise of each model since it encourages each model to select more diverse samples. Decreasing k also helps to encourage inter-model diversity since it allows each sample to be shared by fewer models.

In later stages, the solution of Algorithm 9 becomes more exact. With λ and γ decreasing, according to Lemma 1, the curvature κ_G of $G(\cdot, W)$ approaches 0, the approximation factor $\alpha = 1/2+\kappa_G$ of greedy algorithm increases, and the approximate objective $\hat{H}(W) \geq \alpha H(W)$ becomes closer to the true objective $H(W)$. Moreover, during later stages when the "sample selecting model" constraint ($\mathcal{I}_v$) dominates and λ and γ are almost 0, the inner modular maximization is be exactly solved ($\alpha = 1$) and greedy algorithm degenerates to simple sorting.

The detailed diverse ensemble evolution (DivE^2) procedure is shown in Algorithm 10. The curriculum is composed of T stages. Each stage uses SELECTLEARN (Algorithm 9) to (approximately) solve a continuous-combinatorial optimization in the form of Eq. (5.3) with pre-specified values of (k, p, λ, γ) and initialization $\{w_i^{t-1}\}_{i=1}^m$ from the previous episode as a warm start (line 5). The procedure reduces λ and γ by a multiplicative factor $(1 - \mu)$ in line 6, linearly decreases k by Δ_k and additively increases p by Δ_p, in Line 7. Both k and p are restricted to be integers and within the legal ranges, i.e., $k \in [1, m]$ and $p \in [1, n]$. The warm start initialization is similar in spirit to continuation schemes used in previous curriculum learning (CL) [19, 16, 17, 106, 12, 197, 248] and SPL [116, 207, 201, 208], to avoid getting trapped in local minima and to stabilize optimization. As consecutive problems have the same form with similar parameters (k, p, λ, γ), the solution to the previous problem might still evaluate well on the next one. Hence, instead of running lines 5-14 in

Algorithm 9 until full convergence (as instructed by line 4), we run them for ≤ 10 iterations for reduced running time.

In DivE^2, the parameters k, p, λ, γ defining the learning goal of each stage are gradually change according to a pre-defined schedule. In the future, we plan to use reinforcement learning to train an agent to adaptively select these parameters based some features representing the state of the current learning stage.

In DivE^2, we extend the concept of "machine teaching" to "machine education" by emphasizing the dynamic interaction between teacher (data assignment) and students (models) during the learning process. In machine education, the curriculum is composed of a sequence of learning goals for different learning stages, and each goal (i.e., an optimization in the form of Eq. (5.3)) is achieved based on the current performance of models and the data distribution. In contrast, machine teaching aims at finding the best "teaching set", which is the final goal of the teacher, but does not delicately optimize the learning process. While machine teaching might be useful to convex optimization, machine education that optimizes the learning process (i.e., the curriculum) is more suitable to non-convex optimization for training deep neural networks.

14.4 Appendix

We define $\hat{A}_v \triangleq \hat{A} \cap \delta(v)$ (edges incident to sample v), and $\hat{A}^u \triangleq \hat{A} \cap \delta(u)$ (edges incident to model u) for simplicity.

14.4.1 Proof of Lemma 3

Proof. For the monotone non-decreasing objective $G(\cdot, W)$, greedy algorithm or its variant always tries to add more elements until some constraint(s) is violated. Hence, if the first constraint $\mathcal{I}_v$ does not saturate, i.e., there exists at least one $v' \in V$ such that $|\hat{A}_v| = k - x$ for some integer $1 \leq x \leq m$, and $|\hat{A}^u| = p, \forall u \in U \setminus U(\hat{A}_{v'})$, where $U(\hat{A}_{j'})$ represents the set of models incident to $\hat{A}_{v'}$. That is, for any model u that sample v' is not assigned to, the only reason that it cannot be assigned to sample v' when $|\hat{A}_{v'}| < k$ is that the corresponding second constraint $\mathcal{I}_u$ already saturates. The number of models with saturated constraint $|\hat{A}^u| = p$ is $\left|U \setminus \hat{A}_{v'}\right| = m - k + x$. We then have

$$|\hat{A}| \geq |\hat{A}_{v'}| + \sum_{u \in U \setminus \hat{A}_{v'}} |\hat{A}^u| = (k - x) + (m - k + x)p \geq (k - 1) + (m - k + 1)p. \tag{14.2}$$

For the other $n - 1$ samples excluding sample v', they need to satisfy the constraint $|\hat{A}_v| \leq k$, and they need to be assigned (with repetition) to the $(m - k + x)$ models such that each model gets p samples, i.e.,

$$(n - 1)k \geq \sum_{v \in V \setminus v'} |\hat{A}_v| \geq \sum_{u \in U \setminus \hat{A}_{v'}} |\hat{A}^u| = (m - k + x)p \geq (m - k + 1)p. \tag{14.3}$$

Therefore, if the first constraint does not saturate, we must have

$$k \geq \frac{mp + p}{n + (p - 1)}. \tag{14.4}$$

An equivalent statement due to logical transposition rule is: if $k < {}^{mp+p}/_{n+(p-1)}$, the first constraint must saturate, and $|\hat{A}| = nk$. This completes the proof of the first statement in Lemma 3.

By following similar reasoning, if the second constraint does not saturate, i.e., there exists at least one $u' \in U$ such that $|\hat{A}^{u'}| = p - x$ for certain integer $1 \leq x \leq n$, we have

$$|\hat{A}| \geq |\hat{A}^{u'}| + \sum_{v \in V \setminus \hat{A}^{u'}} |\hat{A}_v| = (p - x) + (n - p + x)k \geq (p - 1) + (n - p + 1)k. \tag{14.5}$$

For the other $m - 1$ models excluding sample u', they need to satisfy the constraint $|\hat{A}^u| \leq p$, and the $(n - p + x)$ samples need to be assigned (with repetition) to these $m - 1$ models such that each sample is assigned to k models, i.e.,

$$(m - 1)p \geq \sum_{u \in U \setminus u'} |\hat{A}^u| \geq \sum_{v \in V \setminus \hat{A}^{u'}} |\hat{A}_v| = (n - p + x)k \geq (n - p + 1)k. \tag{14.6}$$

Hence, if the second constraint does not saturate, we must have

$$k \leq \frac{mp - p}{n - (p - 1)}. \tag{14.7}$$

Similarly, if $k > {}^{mp-p}/_{n-(p-1)}$, the second constraint must saturate, and $|\hat{A}| = mp$. This completes the proof of the second statement in Lemma 3.

By combing the conditions in Eq. (14.4) and Eq. (14.7), and the respective lower bounds of $|\hat{A}|$ in Eq. (14.2) and Eq. (14.5) under these two conditions, the third statement can be proved. □

14.4.2 Proof of Proposition 3

Proof. In each iteration, lines 7-9 in Algorithm 9 guarantees that $G(\hat{A}, W^t) \geq G(A, W^t)$ (where the superscript t refers to the iteration index), and the gradient descent on $-\hat{H}(W^t)$ (or equally gradient ascent on $\hat{H}(W^t)$) guarantees that $H(W^{t+1}) \geq H(W^t)$, which implies $G(\hat{A}, W^{t+1}) \geq G(\hat{A}, W^t)$.

Hence, we have $G(\hat{A}, W^{t+1}) \geq G(\hat{A}, W^t) \geq G(A, W^t)$, i.e., each iteration of Algorithm 9 does not decrease the objective $G(A, W)$ of the max-max problem in Eq. (5.3). So the algorithm generates a monotonically non-decreasing sequence of objective values of $G(A, W)$. This completes the proof of the first statement.

By producing a monotonically non-decreasing sequence of objective values of $G(A, W)$, with a damped learning rate η, Algorithm 9 eventually stays on one piece of $\hat{H}(W)$ and converges to a stationary point on that piece with zero gradient. It will not end by oscillating amongst the non-differentiable boundaries between the pieces on $\hat{H}(W)$ because the algorithm can only visit each boundary point for at most one time due to the monotone non-decreasing objective values. This completes the proof of the second statement.

Given the data assignment $\hat{A}$ produced by lines 6-9 in Algorithm 9, the objective $G(\hat{A}, W)$ can be represented as the sum of $|\hat{A}|$ sample-wise loss functions in the following form.

$$G(\hat{A}, W) = \sum_{u_i \in U} \sum_{v_j \in V(\hat{A}^{u_i})} (\beta - \ell(v_j; w_i)) = \beta|\hat{A}| - \sum_{(v_j, u_i) \in \hat{A}} \ell(v_j; w_i). \tag{14.8}$$

Because each loss function $\ell(v_j; w_i)$ is β-strongly convex w.r.t. w_i, $-G(\hat{A}, W) = -\hat{H}(W)$ is $\beta|\hat{A}|$-strongly convex, which indicates that for any W^*_{loc} and $\hat{W}$,

$$\hat{H}(\hat{W}) + \nabla\hat{H}(\hat{W})^T(W^*_{\text{loc}} - \hat{W}) - \hat{H}(W^*_{\text{loc}}) \geq \frac{\beta|\hat{A}|}{2}\|W^*_{\text{loc}} - \hat{W}\|_2^2. \tag{14.9}$$

Since $-\hat{H}(W)$ is $\beta|\hat{A}|$-strongly convex, any stationary point $\hat{W}$ achieved by Algorithm 9 is a local optimal solution within some local area $\mathcal{K}$. Hence, for any local optimal solution $W^*_{\text{loc}} \in \mathcal{K}$ on the true objective $H(W)$, the above inequality in Eq. (14.9) still holds. In addition, because $\nabla\hat{H}(\hat{W}) = \mathbf{0}$, Eq. (14.9) becomes

$$\hat{H}(\hat{W}) \geq \hat{H}(W^*_{\text{loc}}) + \frac{\beta|\hat{A}|}{2}\|W^*_{\text{loc}} - \hat{W}\|_2^2. \tag{14.10}$$

If SUBMODULARMAX has approximation factor α, we further have $\hat{H}(W^*_{\text{loc}}) \geq \alpha \cdot H(W^*_{\text{loc}})$. Substituting this result and the lower bound for $|\hat{S}|$ from Lemma 3 into Eq. (14.10), we have

$$\hat{H}(\hat{W}) \geq \hat{H}(W^*_{\text{loc}}) + \frac{\beta|\hat{A}|}{2}\|W^*_{\text{loc}} - \hat{W}\|_2^2 \tag{14.11}$$

$$\geq \alpha H(W^*_{\text{loc}}) + \frac{\beta}{2} \cdot \min\{(k-1) + (m-k+1)p, (p-1) + (n-p+1)k\}\|\hat{W} - W^*_{\text{loc}}\|_2^2. \tag{14.12}$$

This completes the proof of the third statement. □

Part IV

APPLICATIONS OF CURRICULUM LEARNING

Curriculum learning is a general learning strategy that can be developed to improve a rich class of machine learning applications. In this part, we apply the curriculum learning algorithms proposed in Part III to supervised learning, weakly supervised learning, and ensemble learning. We evaluate these algorithms on a broad range of benchmarks and compare them with existing training algorithms with or without using curricula. Extensive experiments thoroughly demonstrate the advantages of our proposed curriculum learning approaches.

Chapter 15

SUPERVISED LEARNING

In this chapter, we mainly focus on the applications of three curriculum learning algorithms for supervised learning, i.e., minimax curriculum learning (MCL) proposed in Chapter 9, dynamic instance hardness guided curriculum learning (DIHCL) proposed in Chapter 10, and dynamics-optimized curriculum learning (DoCL) proposed in Chapter 11.

15.1 Minimax Curriculum Learning (MCL)

Method \ Dataset	News20	MNIST	CIFAR10	STL10	SVHN	Fashion
SGD(random)	14.27	0.88	18.52	21.76	5.20	7.79
SPL	15.43	1.25	21.14	20.63	5.67	7.46
SPLD	16.23	1.18	20.79	21.25	5.40	7.80
MCL($\Delta = 0, \lambda = 0, \gamma = 0$)	15.99	1.23	18.04	20.50	5.37	7.95
MCL($\Delta = 0, \lambda > 0, \gamma > 0$)	16.54	0.95	17.33	19.70	4.95	7.29
MCL($\Delta > 0, \lambda > 0, \gamma = 0$)	15.45	0.82	16.93	20.40	5.29	7.07
MCL-RAND	16.23	0.80	17.12	20.42	5.18	6.92
MCL($\Delta > 0, \lambda > 0, \gamma > 0$)	**14.12**	**0.75**	**12.87**	**17.83**	**4.19**	**6.36**

Table 15.1: Test error (%) for different methods (SGD shows the lowest error out of 10 random trials).

In this section, we apply MCL and several curriculum learning methods to train logistic regression models on 20newsgroups [120], LeNet5 models on MNIST [121], convolutional neural nets (CNNs) with three convolutional layers* on CIFAR10 [114], CNNs with two convolutional layers † on Fashion-MNIST ("Fashion" in all tables) [236], CNNs with six convolutional layers on

*The "v3" network from https://github.com/jseppanen/cifar_lasagne.

†A variant of LeNet5 with 64 kernels for each convolutional layer.

STL10 [40], and CNNs with seven convolutional layers on SVHN [161][‡]. Details on the datasets can be found in Table 15.5. In all cases, we also use ℓ_2 parameter regularization on w with weight 1.0^{-4} (i.e., the weight decay factor of the optimizer).

We compare MCL and its variants to SPL [116], SPLD [100] and SGD with a random curriculum (i.e., with random batches). Each method uses mini-batch SGD for $\pi(\cdot, \eta)$ with the same learning rate strategy to update w. The methods, therefore, differ only in the curriculum (i.e., the sequence of training sets).

15.1.1 Settings and Hyperparameters

For SGD, in each iteration, we randomly select 4000 samples (20newsgroups) or 5000 samples (other datasets) and apply mini-batch SGD to the selected samples. In SPL and SPLD, the training set starts from a fixed size k (4000 samples for 20newsgroups, 5000 samples for other datasets), and increases by a factor of $1 + \mu$ (where $\mu = 0.1$) per round of alternating minimization (i.e., per iteration of the outer loop) [§]. We use ρ to denote the number of iterations of the inner loop, which aims to minimize the loss w.r.t. the model w on the selected training set. In SPLD, we also have a weight for the negative group sparsity: it starts from ξ and increases by a factor of 1.1 at each round of alternating minimization (i.e., per iteration of the outer loop). We test five different combinations of $\{\rho, \mu\}$ and $\{\rho, \xi\}$ for SPL and SPLD respectively. The best combination with the smallest test error rate is what we report. Neither SPL nor SPLD uses the clustering trick we applied to MCL: they compute the exact loss on each sample in each iteration. Hence, they have more accurate estimation of the hardness on each sample, and require knowing the labels of all samples (selected and unselected) and cannot reduce annotation costs. Note SPLD still needs to run clustering and use

[‡]The network structures for STL10 and SVHN can be found at https://github.com/aaron-xichen/pytorch-playground

[§]Similar to [100], instead of specifying the absolute value of λ in each iteration, we find that specifying the number of selected samples k is more robust empirically. Because directly setting λ can result in selecting too many or too few samples, but SPL/SPLD needs to increase training samples gradually.

the resulted clusters as groups in the group sparsity (which measures diversity in SPLD). We did not select samples with SPL/SPLD as we do with MCL since we wanted to test SPL/SPLD as originally presented — intuitively, SPL/SPLD should if anything only do better without such clustering due to the more accurate sample-specific hardness estimation. The actual clustering, however, used for SPLD's diversity term is the same as that used for MCL's cluster samples. We apply the mini-batch k-means algorithm to the features detailed in the next paragraph to get the clusters used in MCL and SPLD. Although both SPL and SPLD can be reduced to SGD when $\lambda \to \infty$ (i.e., all samples always selected), we do not include this special case because SGD is already a baseline. For SGD with a random curriculum, results of 10 independent trials are reported.

Dataset	News20	MNIST	CIFAR10	STL10	SVHN	Fashion
Total time	2649.19s	3418.97s	3677.73s	2953.47s	34153.81s	2927.18s
WS-SUBMODULARMAX	62.44s	35.33s	127.36s	206.70s	1892.62s	167.55s

Table 15.2: Total time (secs.) of MCL($\Delta > 0, \lambda > 0, \gamma > 0$) and time only on WS-SUBMODULARMAX.

In our MCL experiments, we use a simple "feature based" submodular function [231] where $F(A) = \sum_{u \in \mathcal{U}} \omega_u \sqrt{c_u(A)}$ and where $\mathcal{U}$ is a set of features. For a subset A of clusters, $c_u(A) = \sum_{i \in A} c_u(i)$, where $c_u(i)$ is the nonnegative feature u of the centroid for cluster i, and can be interpreted as a nonnegative score for cluster i. We use TF-IDF features for 20newsgroup. For the other datasets, we train a corresponding neural network on a small random subset of training data (e.g., hundreds of samples) for one epoch, and use the inputs to the last fully connected layer (whose outputs are processed by softmax to generate class probabilities) as features. Because we always use ReLU activations between layers, the features are all nonnegative and the submodularity of $F(A)$ follows as a consequence. These features are also used by mini-batch k-means to generate clusters for MCL and SPLD.

For MCL, we set the number of inner loop iterations to $p \leq 50$. For each dataset, we choose p

as the number among $\{10, 20, 50\}$ that reduces the training loss the most in the first few iterations of the outer loop, and then use that p for the remaining iterations. As shown in Table 15.6, we use $p = 50$ for 20newsgroups, MNIST and Fashion-MNIST, and $p = 20$ for the other three datasets.

We consider five variants of MCL: (1) MCL($\Delta = 0$, $\lambda = 0$, $\gamma = 0$) having neither submodular regularization that promotes diversity nor scheduling of k that increases hardness; (2) MCL($\Delta = 0$, $\lambda > 0$, $\gamma > 0$), which decreases diversity by exponentially reducing the weight λ of the submodular regularization, but does not have any scheduling of k, i.e., k is fixed during the algorithm; (3) MCL($\Delta > 0$, $\lambda > 0$, $\gamma = 0$), which only uses the scheduling of k shown in Algorithm 2, but the diversity weight λ is positive and fixed during the algorithm, i.e., with $\gamma = 0$; (4) MCL-RAND(r,q), which randomly samples r clusters as a training set $\hat{A}$ after every q rounds of the outer loop in Algorithm 2, and thus combines both MCL and SGD; (5) MCL($\Delta > 0$, $\lambda > 0$, $\gamma > 0$), which uses the scheduling of both λ and k shown in Algorithm 2. We tried five different combinations of $\{q, r\}$ for MCL-RAND(r,q) and five different Δ values for MCL($\Delta > 0$, $\lambda > 0$, $\gamma > 0$), and report the one with the smallest test error. Other parameters, such as the initial values for λ and k, the values for γ and p, and the total number of clusters are the same for different variants (the exact values of these quantities are given in Table 15.6).

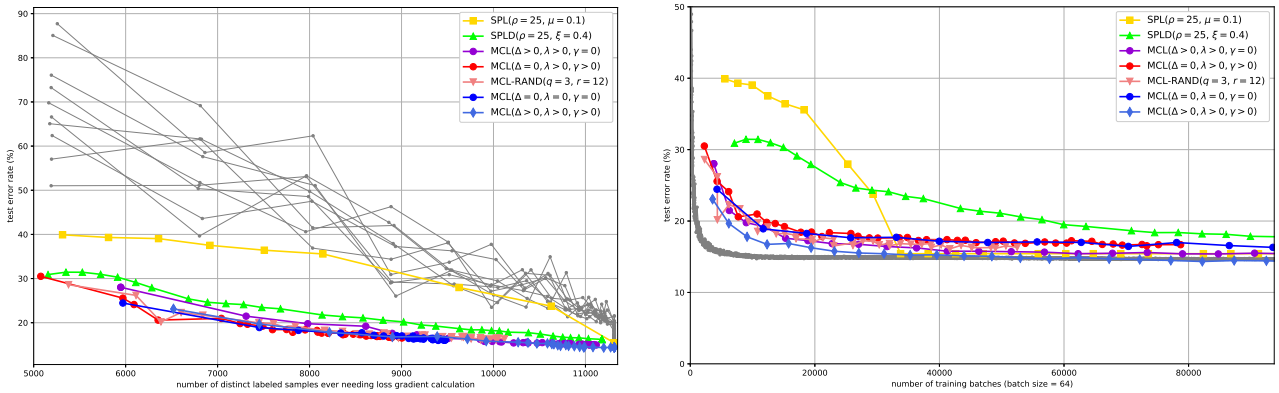

Figure 15.1: Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD).

15.1.2 Main Experimental Results

We summarize the main results in Figure 15.1-15.8. More results are given in Section 15.4.2. In all figures, grey curves correspond to the ten trials of SGD under a random curriculum. The legend in all figures gives the parameters used for the different methods using the following labels: (1) SPL (ρ, μ); (2) SPLD(ρ, ξ); and (3) MCL-RAND(q, r).

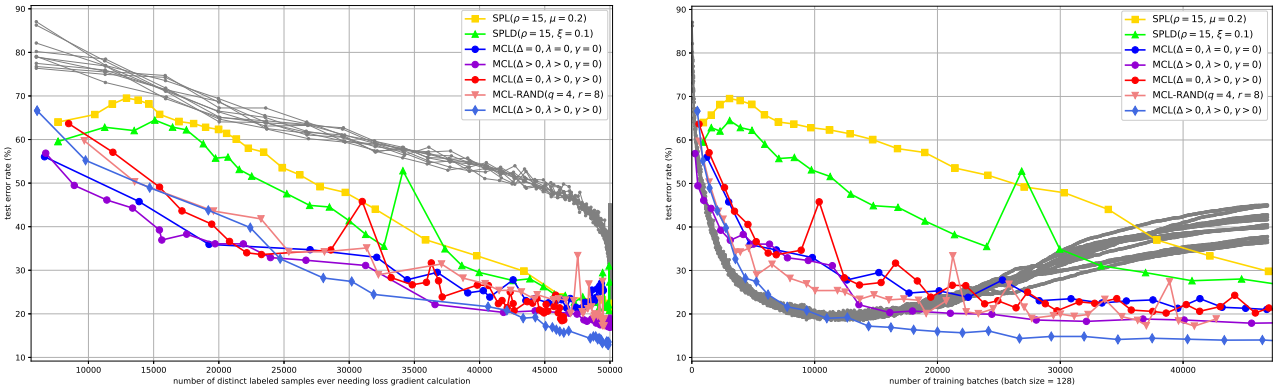

Figure 15.2: Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on CIFAR10 (grey curves represents 10 random trials of SGD).

Figures 15.1-15.6 show how the test error changes with (on the left) the number of distinct labeled samples ever needing a loss gradient calculation, and (on the right) the number of training batches, corresponding to training time. Note only MCL and its variants use the clustering trick, while SPL/SPLD need to compute loss on every sample and thus require knowledge of the labels of all samples. The left plot shows only the number of loss gradient calculations needed — (1) in MCL, for those clusters never selected in the curriculum, the loss (and hence the label) of only the centroid sample is needed; (2) in SPL/SPLD, for those samples never selected in the curriculum, their labels are needed only to compute the loss but not the gradient, so they are not reflected in the left plots of all figures because their labels are not used to compute a gradient. Therefore, thanks to the clustering trick, MCL and its variants can train without needing all labels, similar to semi-supervised learning

methods. This can help to reduce the annotation costs, if an MCL process is done in tandem with a labeling procedure analogous to active learning. The right plots very roughly indicate convergence rate, namely how the test error decreases as a function of the amount of training.

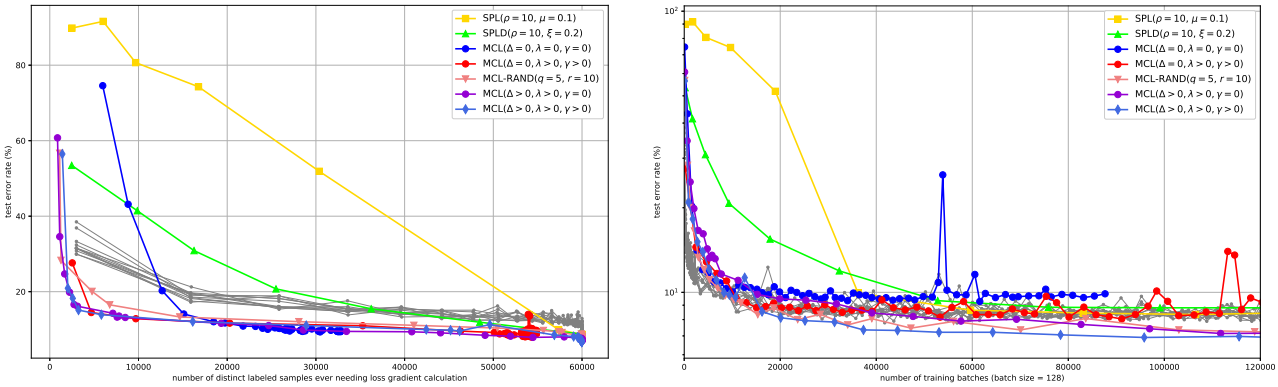

Figure 15.3: Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on Fashion-MNIST (grey curves represents 10 random trials of SGD).

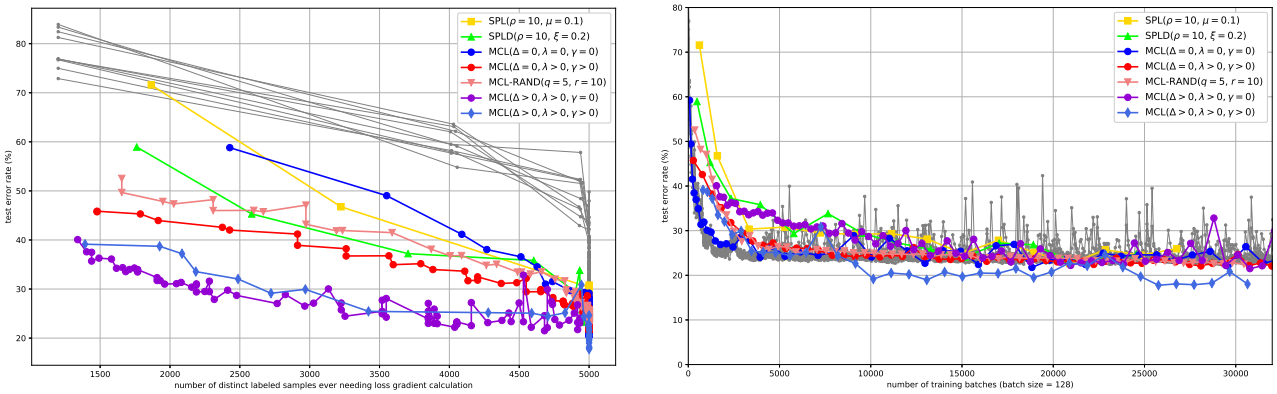

Figure 15.4: Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on STL10 (grey curves represents 10 random trials of SGD).

On all datasets, MCL and most of its variants outperform SPL and SPLD in terms of final test accuracy (shown in Table 15.1) with comparable efficiency (shown in the right plots of all figures). MCL is slightly slower than SGD to converge in early stages but it can achieve a much smaller error

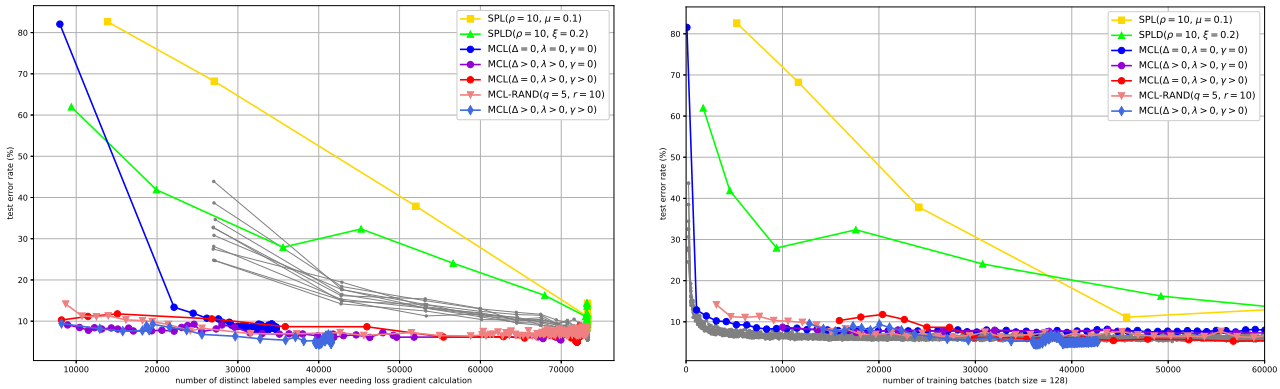

Figure 15.5: Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on SVHN (grey curves represents 10 random trials of SGD).

when using the same number of labeled samples for loss gradients. Moreover, when using the same learning rate strategy, they can be more robust to overfitting, as shown in Figure 15.2. Comparing Figure 15.1 with Figure 15.2-15.6, MCL has the advantage when applied to deep models.

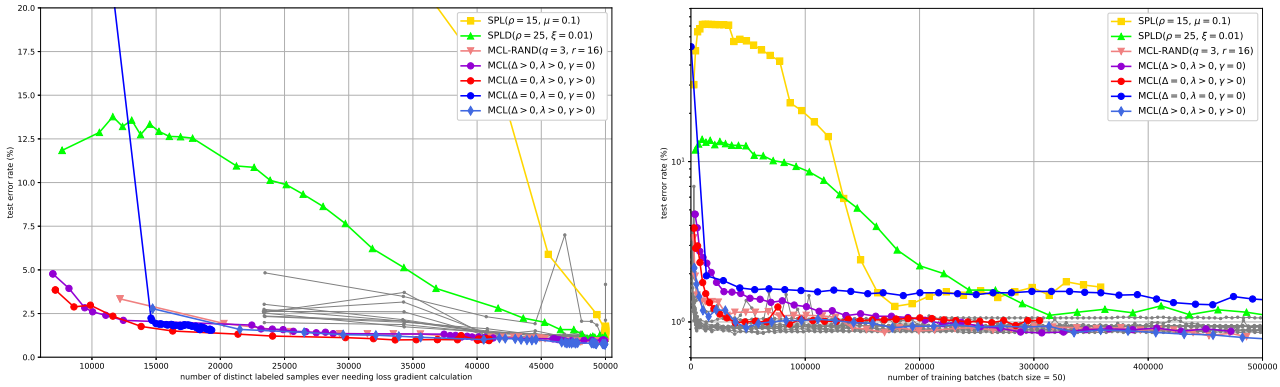

Figure 15.6: Test error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on MNIST (grey curves represents 10 random trials of SGD).

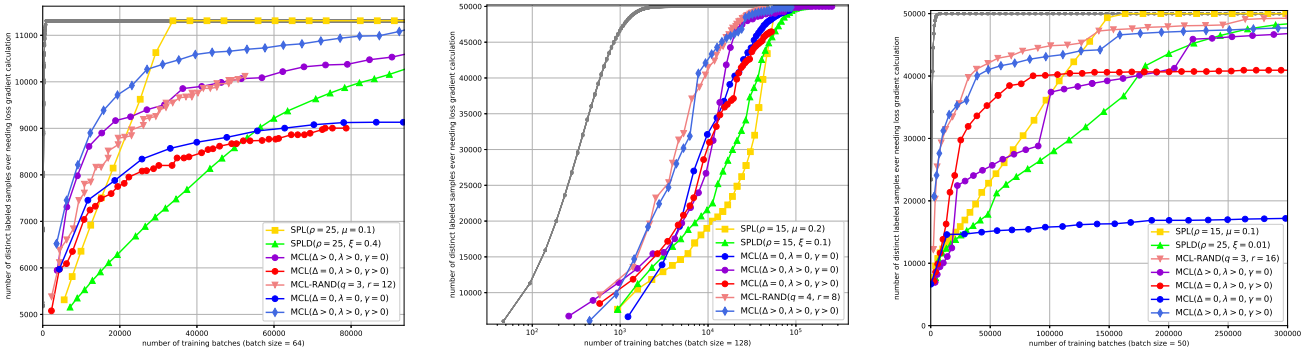

Figure 15.7: Number of distinct labeled samples ever needing loss gradient calculation vs. number of training batches for News20 (left), CIFAR10 (middle) and MNIST(right) (grey curves represents 10 random trials of SGD).

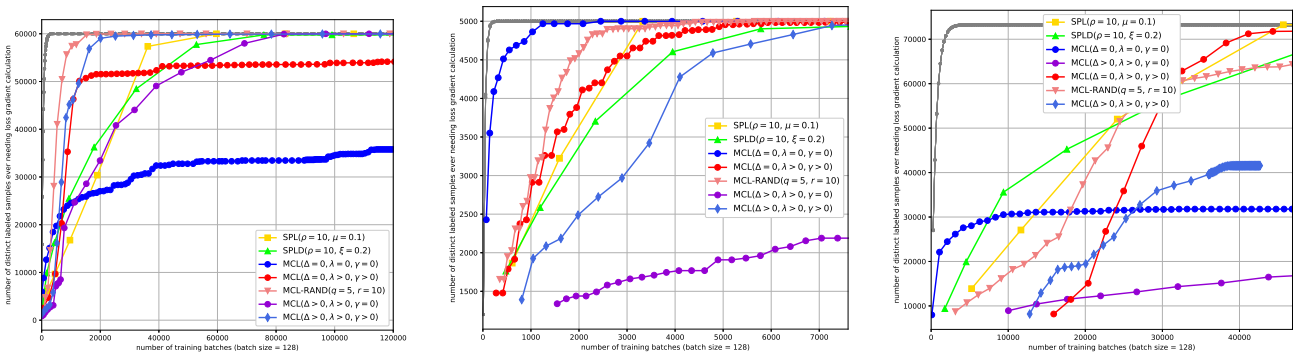

Figure 15.8: Number of distinct labeled samples ever needing loss gradient calculation vs. number of training batches for Fashion-MNIST (left), STL10 (middle) and SVHN (right) (grey curves represents 10 random trials of SGD).

15.1.3 Efficiency

In MCL, running greedy is the only extra computation comparing to normal SGD. To show that in our implementation (see Section 9.3.1) its additional time cost is negligible, we report in Table 15.2 the total time cost for MCL($\Delta > 0$, $\lambda > 0$, $\gamma > 0$) and the time spent on our implementation WS-SUBMODULARMAX.

Figure 15.7 and Figure 15.8 shows how the "number of distinct labeled samples ever needing loss gradient calculation" changes as training proceeds. It shows how the different methods trade-

off between "training on more new samples" vs. "training on fewer distinct samples more often." Thanks to the clustering trick, MCL and its variants usually require fewer labeled samples for model training than SGD but more than SPL and SPLD.

15.1.4 Ablation Study

Among the five variants of MCL, MCL($\lambda > 0, \gamma > 0, \Delta > 0$) achieves the fastest convergence speed in later stages and the smallest final test error, while MCL($\Delta = 0, \lambda = 0, \Delta = 0$) usually achieves the worst performance (the only exception is on News20). Comparison between MCL($\Delta = 0$, $\lambda > 0, \gamma > 0$) and MCL($\lambda = 0, \gamma = 0, \Delta = 0$) shows that decreasing diversity improves the performance. MCL($\lambda > 0, \gamma > 0, \Delta > 0$) always outperforms MCL($\Delta = 0, \lambda > 0, \gamma > 0$). This indicates that increasing k can bring advantages, e.g., more improvements in later stages.

MCL($\lambda > 0, \gamma > 0, \Delta > 0$) always outperforms MCL($\Delta > 0, \lambda > 0, \gamma = 0$), which supports our claim that it is better to decrease the diversity as training proceeds rather than keeping it fixed. In particular, MCL($\Delta > 0, \lambda > 0, \gamma = 0$) shows slower convergence than other MCL variants in later stages. In our experiments in the MCL($\Delta > 0, \lambda > 0, \gamma = 0$) case, we needed to carefully choose λ and use a relatively large Δ for it to work at all, as otherwise it would repeatedly choose the same subset (with small Δ, the loss term decreases as training proceeds, so with fixed λ the diversity term comes to dominate the objective). This suggests that a large diversity encouragement is neither necessary nor beneficial when the model matures, possibly since k is large at that point and there is ample opportunity for a diversity of samples to be selected just because k is large, and also since encouraging too much loss-unspecific diversity at that point might only select outliers.

The combination of MCL and random curriculum (MCL-RAND) speeds up convergence, and sometimes (e.g., on MNIST, SVHN and Fashion-MNIST) leads to a good final test accuracy, but requires more labeled samples for gradient computation and still cannot outperform MCL($\lambda > 0$, $\gamma > 0, \Delta > 0$). These results indicate that the diversity introduced by submodular regularization

does yield improvements, and changing both hardness and diversity improves performance.

15.2 Dynamic Instance Hardness guided Curriculum Learning (DIHCL)

We train different DNNs by using variants of DIHCL, and compare them with three baselines, vanilla random mini-batch SGD, self-paced learning (SPL) [116], and minimax curriculum learning (MCL) [248] on 11 image classification datasets (without pre-training), i.e., **(A)** WideResNet-28-10 [240] on CIFAR10 and CIFAR100 [114]; **(B)** ResNeXt50-32x4d [238] on Food-101 [27], FGVC Aircraft (Aircraft) [146], Stanford Cars [113], and Birdsnap [20]; **(C)** ResNet50 [81] on ImageNet [51]; **(D)** WideResNet-16-8 on Fashion-MNIST (FMNIST) [236] and Kuzushiji-MNIST (KMNIST) [39]; **(E)** PreActResNet34 [81] on STL10 [40] and SVHN [161].

15.2.1 Hyperparameters

We use mini-batch SGD with a momentum of 0.9 and a cyclic cosine annealing learning rate schedule [142] (multiple epochs with starting/target learning rate decayed by a multiplicative factor 0.85). We use $T_0 = 5, \gamma = 0.95, \gamma_k = 0.85$ for all DIHCL variants, and gradually reduce k from n to $0.2n$. We chose $T_0 = 5$ since it is sufficient to produce a reasonably good model to estimate DIH. We tried $\gamma = 0.95, 0.9, 0, 8$ and they perform similarly, e.g., for DIHCL-Rand Loss, on CIFAR10, $\gamma = 0.95, 0.9, 0, 8$ lead to accuracy of $96.76\%, 96.75\%, 96.78\%$ respectively. We chose $\gamma_k = 0.85$ so we can reduce the size of S_t from n to $0.2n$ in 10 epochs. On each dataset, we apply each method to train the same model for the same number of epochs, but each method may select a different number of samples per epoch. More details about the datasets and settings can be found in Section 15.4.1. For DIHCL variants that further reduce S_t by solving Eq. (10.2), we use $\lambda_1 = 1.0, \gamma_\lambda = 0.8, \gamma_{k'} = 0.4$ and employ the "facility location" submodular function [42] $G(S) = \sum_{j \in S_t} \max_{i \in S} \omega_{i,j}$ where $\omega_{i,j}$ represents the similarity between sample x_i and x_j. We utilize a Gaussian kernel for similarity using neural net features (e.g., the inputs to the last fully

connected layer in our experiments) $z(x)$ for each x, i.e., $\omega_{i,j} = \exp\left(-\|z(x_i)-z(x_j)\|^2/2\sigma^2\right)$, where σ is the mean value of all the $k(k-1)/2$ pairwise distances.

15.2.2 Main Experimental Results

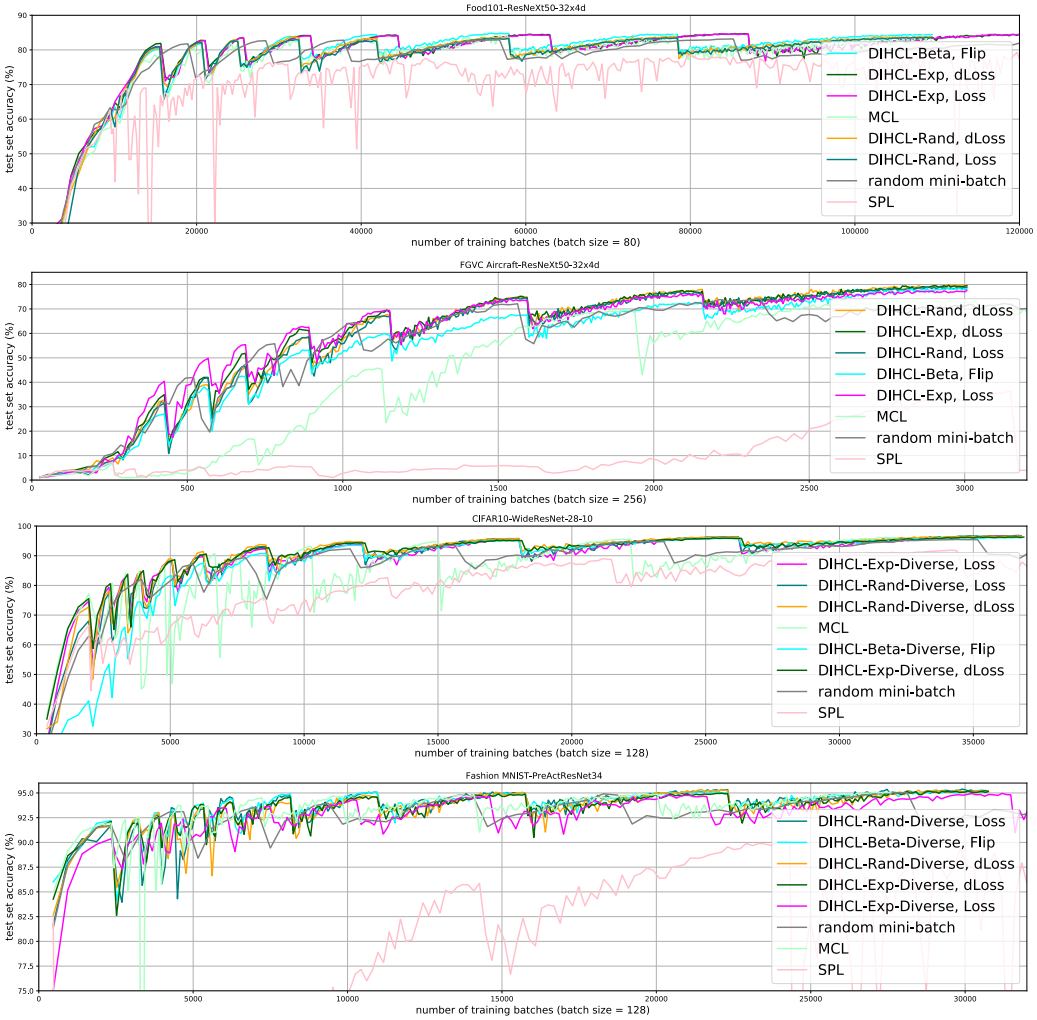

Figure 15.9: Training DNNs by using DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch SGD on Food-101, Aircraft, CIFAR-10 and FMNIST. We use "Diverse" to denote DIHCL that further reduces S_t by applying submodular maximization for Eq. (10.2). We report how test accuracy changes during training.

In Figure 15.9, we show how the test set accuracy changes when increasing the number of training batches in each curriculum learning method on 4 datasets. In Figure 15.10, we report wall-clock

Table 15.3: The test accuracy (%) achieved by different methods training DNNs on 11 datasets (without pre-training). We use "Loss, dLoss, Flip" to denote the 3 choices of DIH metrics based on (A), (B), and (C) respectively. In all DIHCL variants, we apply diversity (and greedy submodular maximization using the lazier-than-lazy-greedy procedure [153]) for Eq. (10.2) on only the datasets CIFAR10, CIFAR100, STL10, SVHN, KMNIST, and FMNIST. In this case, the first T_0 warm-start epochs of DNN training was used to also produce the feature extractor $z(x)$ to instantiate the facility location function. The other datasets did not employ diversity, and we leave that to future work. For each dataset, the best accuracy is in blue, the second best is red, and third best green.

Curriculum	CIFAR10	CIFAR100	Food-101	ImageNet	STL10	SVHN	KMNIST	FMNIST	Birdsnap	Aircraft	Cars
Rand mini-batch	96.18	79.64	83.56	75.04	86.06	96.48	98.67	95.22	64.23	74.71	78.73
SPL	93.55	80.25	81.36	73.23	81.33	96.15	97.24	92.09	63.26	68.95	77.61
MCL	96.60	80.99	84.18	75.09	88.57	96.93	99.09	95.07	65.76	75.28	76.98
DIHCL-Rand, Loss	96.76	80.77	83.82	75.41	87.25	96.81	99.10	95.69	65.62	79.00	80.91
DIHCL-Rand, dLoss	96.73	80.65	83.82	75.34	86.93	96.83	99.14	95.64	65.25	79.93	78.70
DIHCL-Exp, Loss	97.03	82.23	84.65	75.10	88.36	96.91	99.20	95.45	66.13	77.68	79.85
DIHCL-Exp, dLoss	96.40	81.42	84.75	75.62	89.41	96.80	99.18	95.50	66.59	79.72	81.48
DIHCL-Beta, Flip	96.51	81.06	84.94	76.33	86.88	97.18	99.05	95.66	65.48	78.49	80.13

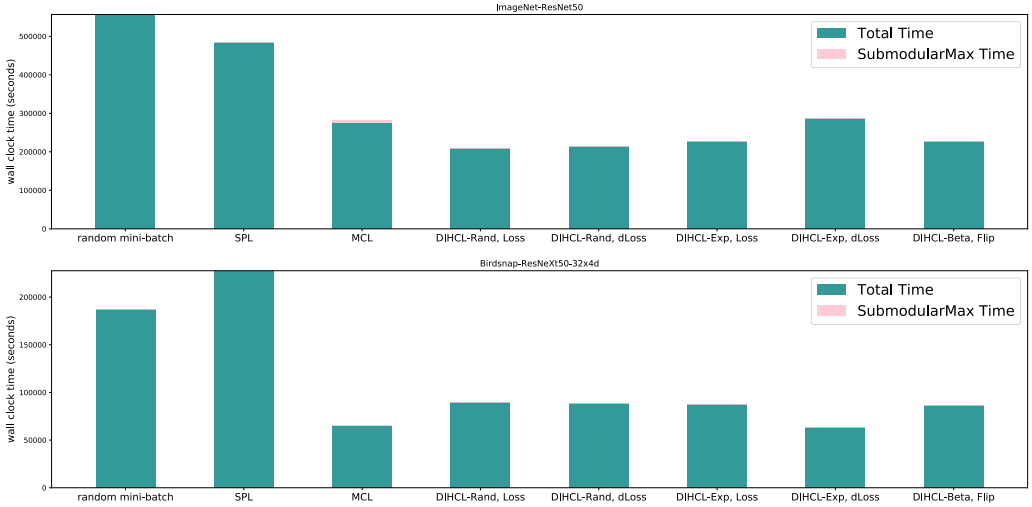

Figure 15.10: Wall clock training time of DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch on ImageNet (top) and Food-101 (bottom).

training time on 2 datasets. *The results for other datasets can be found in Section 15.4.4*, together with the wall-clock time for (1) the entire training and (2) the submodular maximization part in DIHCL with diversity and MCL. The final test accuracy achieved by each method is reported in Table 15.3. DIHCL and its variants show significantly faster and smoother gains on test accuracy than baselines during training especially at earlier stages. DIHCL also achieves higher final accuracy and shows improvements in sample efficiency (meaning they reach their best performance sooner, after less computation has taken place). MCL can reach similar performance as DIHCL on some datasets

but it shows less stability and requires more relative computation for submodular maximization. We also observe a similar instability of SPL. The reason is that, compared to the methods that use DIH, both MCL and SPL deploy instantaneous instance hardness (i.e., current loss) as the score to select samples, a measure that is more sensitive to randomness and perturbation that occurs during training. Compared to MCL and DIHCL, SPL and the random mini-batch curriculum method requires more epochs to reach their best accuracy, since they spend training effort on the memorable samples but lack repeated-learning of the forgettable ones. Although every variant of DIHCL achieves the best accuracy among all the evaluated methods on some datasets, DIHCL-Exp using loss and DIHCL-Beta using prediction flips, as the instantaneous hardness, exhibit advantages over the other DIHCL variants. Particularly, DIHCL-Exp with dLoss(metric (B)) is the best variant across datasets (achieving the top-2 performance on 8 out of the 11 datasets).

15.2.3 Ablation Study

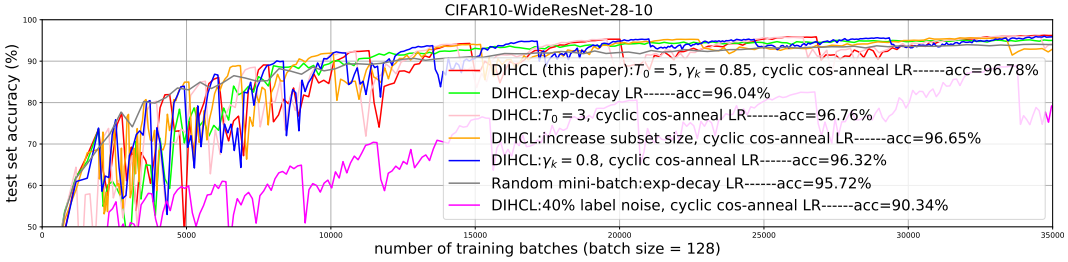

Figure 15.11: Comparison of DIHCL variants on training WideResNet-28-10 on CIFAR10.

We conduct an ablation study comparing several possible variants of DIHCL with their results reported in Figure 15.11. Specifically, we (1) change the cyclic cosine annealing learning rate to more commonly used exponentially decaying learning rate and compare DIHCL with random mini-batches; (2) reduce the number of warm starting epochs T_0 from 5 to 3; (3) increase the budget k instead of decrease it in Line 12 of Algorithm 4 using $\gamma_k = 1.2$; (4) use a smaller discounting factor $\gamma_k = 0.8$; or (5) apply DIHCL on dataset containing 40% noisy (wrong) labels. The results

Table 15.4: The test accuracy (%) achieved by random mini-batch SGD (Random), SPL, MCL, DoCL-NR (DIHCL) and DoCL in training DNNs on 9 datasets (without pre-training). In MCL, DoCL-NR and DoCL, we apply lazier-than-lazy-greedy [153] for Eq. (11.1) on CIFAR10, CIFAR100, SVHN and FMNIST. DoCL achieves the highest test accuracy over all 9 datasets.

Curriculum	CIFAR10	CIFAR100	Food-101	ImageNet	SVHN	FMNIST	Birdsnap	Aircraft	Cars
Random	96.18	79.64	83.56	75.04	96.48	95.22	64.23	74.71	78.73
SPL [116]	93.55	80.25	81.36	73.23	96.15	92.09	63.26	68.95	77.61
MCL [248]	96.60	80.99	84.18	75.09	96.93	95.07	65.76	75.28	76.98
DoCL-NR	96.40	81.42	84.75	75.62	96.80	95.50	66.59	79.72	81.48
DoCL (Ours)	**97.43**	**83.23**	**87.45**	**79.54**	**97.36**	**95.89**	**71.37**	**82.40**	**86.26**

show that original DIHCL outperforms all the variants and is robust to noisy labels.

15.3 Dynamics-optimized Curriculum Learning (DoCL)

We compare DoCL with the widely-used random mini-batch SGD, two recent curriculum learning methods, i.e., self-paced learning (SPL) [116] and minimax curriculum learning [248], and one variant of DoCL (DoCL-NR) by using them to train different DNNs architectures on 9 image classification datasets (without pre-training), i.e., **(A)** WideResNet-28-10 [240] on CIFAR10 and CIFAR100 [114]; **(B)** ResNeXt50-32x4d [238] on Food-101 [27], FGVC Aircraft (Aircraft) [146], Stanford Cars [113], and Birdsnap [20]; **(C)** ResNet50 [81] on ImageNet [51]; **(D)** WideResNet-16-8 on Fashion-MNIST (FMNIST) [236]; **(E)** PreActResNet34 [81] on SVHN [161]. The major difference across different curriculum learning methods lies in the criteria to select samples. Random mini-batch SGD adopts the criterion of uniform sampling over training set. SPL and MCL rely on the instantaneous loss of each sample: SPL tends to select easier samples while MCL prefers harder ones and applies an additional diversity criterion as in Eq. (11.1). As defined in Eq. (8.4) and Eq. (8.7), DoCL's criteria are built upon the inner product of the residuals and linear dynamics. Hence, MCL can also be seen as a variant of DoCL that only considers the instantaneous feedback on the residual part. To complete this ablation study, we consider another variant DoCL-NR (NR stands for "no

residual") that only relies on the linear dynamics part, i.e., it instead uses $a_t(i) = \|\partial f(x_i;\theta_t)/\partial t\|_2$ to compute the running mean in Eq. (8.5) and follows the schedule of increasing samples over epochs as MCL (vs. decreasing samples in DoCL). One may notice that DoCL-NR is actually DIHCL [253] using the linear dynamics $a_t(i) = \|\partial f(x_i;\theta_t)/\partial t\|_2$ as the instantaneous feedback that it applies lazy exponential moving averaging to. Therefore, the ablation study also can be seem as a comparison between DoCL and DIHCL.

15.3.1 Hyperparameters

For all these methods, we update the model on their selected/sampled data using mini-batch SGD with momentum of 0.9. We use a cyclical cosine annealing learning rate schedule [142] (multiple cycles with ending epoch numbers $\{\eta_t\}_{t=1}^{T_\kappa}$ and with starting/target learning rate decayed by a multiplicative factor 0.85). It will suffer a short period of accuracy drop due to the surging learning rate at the beginning of every cycle (as in Figure 15.12-15.13) but can traverse more regions with different local minima on the loss landscape and eventually achieve better performance with faster long-term convergence. On each dataset, we tried a handful of schedules on mini-batch SGD, chose the one with the best validation accuracy, and used it for all the methods (but each method may select different numbers of samples per epoch). We apply standard data augmentation on all datasets and Mix-up [244] with $\alpha = 0.4$. More details about the datasets, options of $\{\eta_t\}_{t=1}^{T_\kappa}$ and other training hyperparameters shared across methods can be found in Section 15.4.5. In DoCL-NR and DoCL, we set $\gamma_k = \gamma_\lambda = 0.9, k_{\min} = 0.2n$. We tried a few common choices for them, e.g., $\gamma_k, \gamma_\lambda \in \{0.85, 0.9, 0.95\}$ and chose the best one. On some datasets, we further test the performance of Eq. (11.1) in reducing S_t and employ the "facility location" submodular function [42] $G(S) = \sum_{j \in S_t} \max_{i \in S} \omega_{i,j}$ as a diversity criterion, where $\omega_{i,j}$ represents the similarity between sample x_i and x_j. We utilize a Gaussian kernel for similarity measurement using neural net features (i.e., the inputs to the last fully connected layer in our experiments) $z(x)$ for each x, i.e.,

$\omega_{i,j} = \exp\left(-{\|z(x_i)-z(x_j)\|^2}/{2\sigma^2}\right)$, where σ is the mean value of all the ${k(k-1)}/{2}$ pairwise distances.

15.3.2 Main Experimental Results

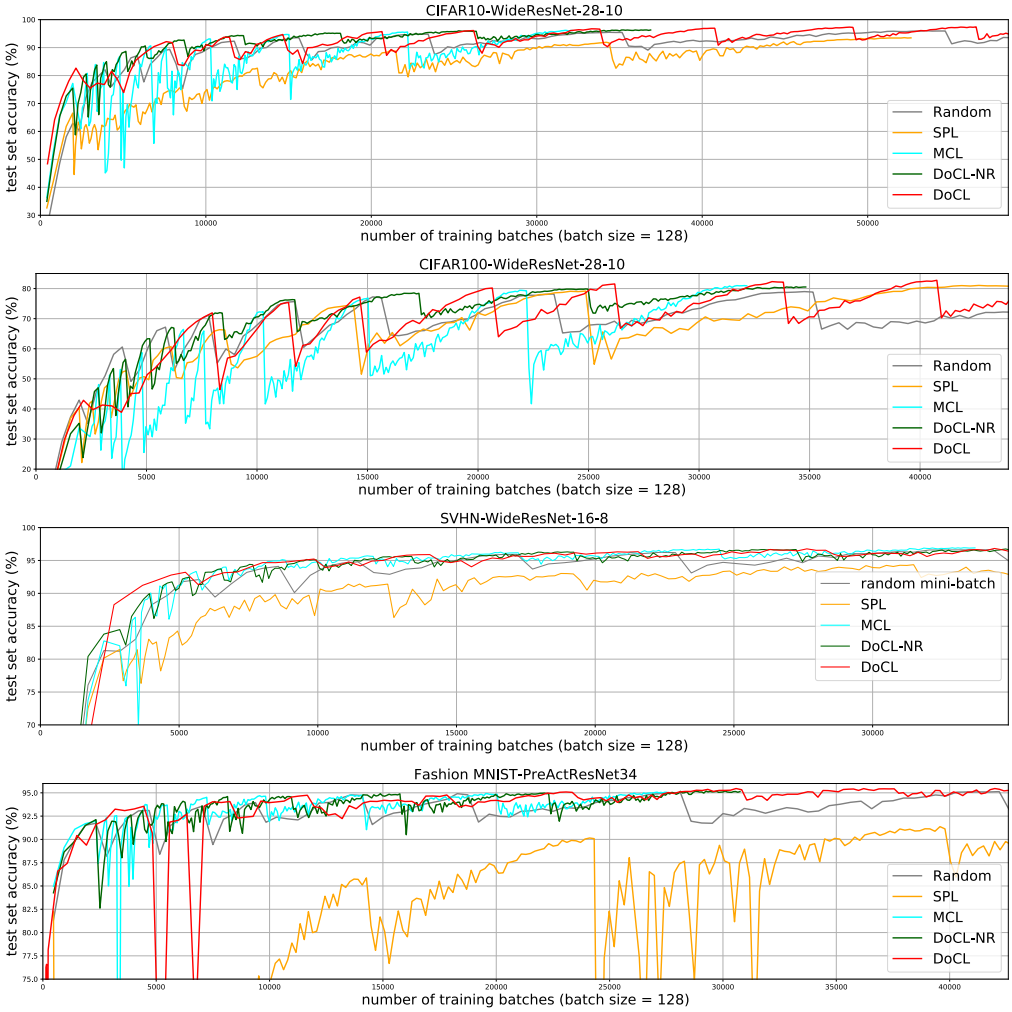

Figure 15.12: Training DNNs by using DoCL, DoCL-NR (DIHCL), SPL [116], MCL [248], and random mini-batch SGD on 4 datasets, i.e., CIFAR10, CIFAR100, SVHN and Fashion MNIST. We report how the test accuracy changes with the number of training batches for each method.

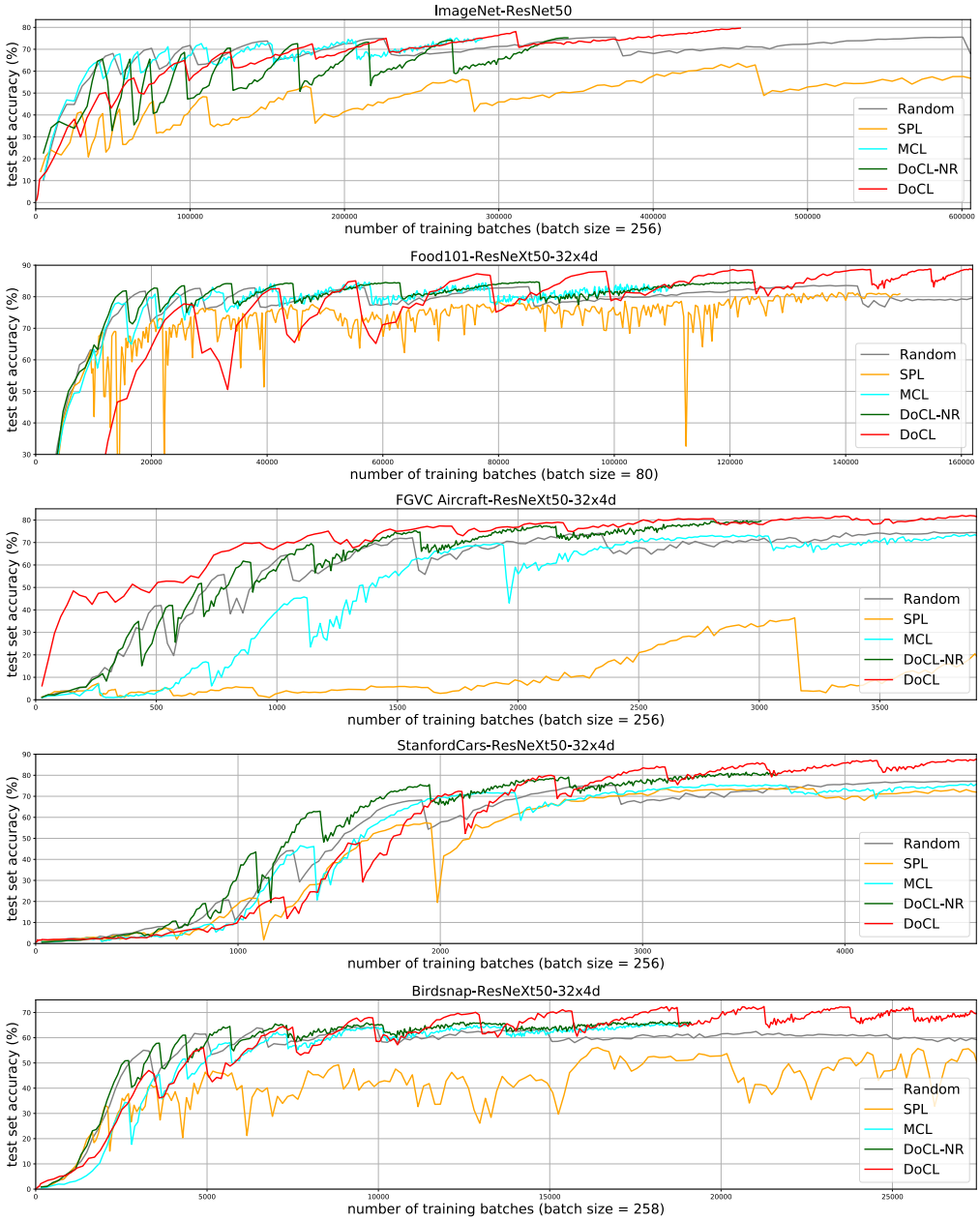

Figure 15.13: Training DNNs by using DoCL, DoCL-NR (DIHCL), SPL [116], MCL [248], and random mini-batch SGD on 5 datasets, i.e., ImageNet, Food101, FGVC Aircraft, Stanford Cars and Birdsnap. We report how the test accuracy changes with the number of training batches for each method.

In Table 15.4, we summarize the final test accuracy achieved by every method on all the 9 datasets.

DoCL achieves the highest test accuracy among all the evaluated methods and outperforms them by

a large margin. On the four fine-grained classification datasets (i.e., Food101, Birdsnap, Aircraft and Cars) that traditional solutions usually rely on fine tuning a pre-trained model, DoCL significantly improves the training from scratch so the resulting accuracy can be comparable with the fine-tuned models. Furthermore, DoCL improves the state-of-the-art top-1 accuracy of ResNet50 on ImageNet from 79.29% [82] (after applying a bag of tricks that might be specific for ImageNet) to 79.54% without heavy tuning of tricks and hyperparameters.

15.3.3 Efficiency

In Figure 15.12-15.13, we report how the test accuracy improves with the increasing number of training batches on nine datasets. During the first several cycles, DoCL has lower accuracies than some other baselines because it starts from the whole training set and gradually reduces the samples per epoch (i.e., line 20 of Algorithm 6) while other CL methods increase the samples per epoch from a small number. DoCL selects samples in decreasing numbers since it needs sufficient exploration of more samples in line with the fast-changing linear dynamics during earlier stages. Hence, given the same number of epochs, DoCL has longer cycles than others. As the cycle length decreases for later stages/cycles, the optimization of dynamics in DoCL prevails and improves the test accuracy much faster than other methods.

15.3.4 Ablation Study

In Figure 15.14, we conduct a thorough ablation study of DoCL (i.e., removing diversity pruning in line 10 of Algorithm 6, or changing the cyclical learning rate to exponential decaying learning rate over episodes) and compare the default hyperparameters (i.e., $T_0 = 5, \tau = 0.1, \gamma = 0.9$) used in above experiments with other options. It shows that DoCL is not very sensitive to most hyperparameters. Specifically, DoCL equipped with diversity pruning, cyclical learning rates, longer warm-starting epochs, lower temperature τ, and moderate γ works slightly better than their

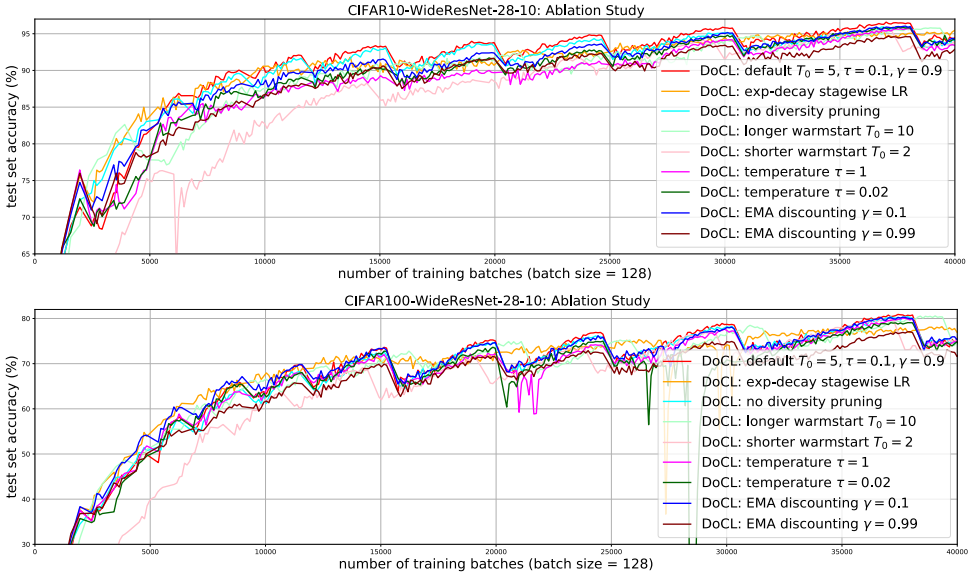

Figure 15.14: Ablation study and sensitivity analysis of hyperparameters for DoCL when applied to train WideResNet-28-10 on CIFAR10 (top) and CIFAR100 (bottom).

counterparts. DoCL with a very short warm-starting period (2 epochs) suffers from insufficient exploration over all samples in earlier stages and thus performs much poorer than other variants but it finally achieve similar test accuracy as others in later stages.

15.3.5 Regression

Although the above experiments mainly focus on classification tasks, we also evaluate the performance of DoCL and compare it with other baselines on a regression task called "knowledge distillation" [30, 180, 83] that aims to transfer the knowledge of a pre-trained large neural net to a smaller one (e.g., ResNet-18). In particular, we study an L2 regression minimizing the L2 loss between the output logits (the last-layer outputs before softmax) of a ResNet-18 model and the logits produced by a pre-trained ResNeXt-29 8x64d on the same data. We apply different methods to sequentially select the data subset in each epoch on which we minimize the L2 distillation loss. In Figure 15.15, we report their performance on CIFAR10 and CIFAR100, which shows that DoCL

achieves the best test accuracy among all the methods. On CIFAR100, DoCL keeps outperforming the others starting at very early stages. On CIFAR10, DoCL improves slower during earlier stages due to the decreasing schedule of subset size adopted by DoCL (as illustrated before) but it surpasses others after 30,000 training batches.

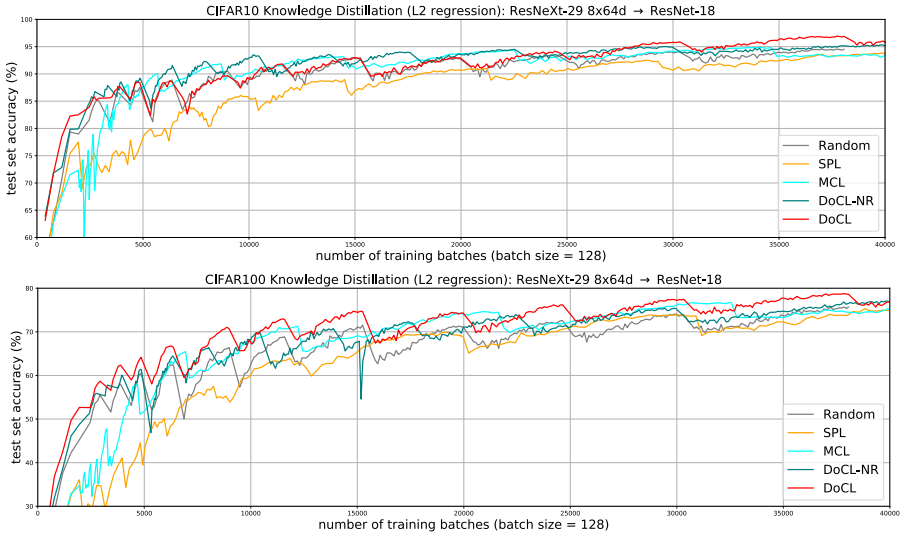

Figure 15.15: Regression (knowledge distillation with L2 loss on pre-softmax logits) by DoCL, DoCL-NR, SPL [116], MCL [248], and random mini-batch SGD for transferring the knowledge of a pre-trained ResNeXt-29 8x64d (34.4M parameters) to ResNet-18 (11.2M parameters) on CIFAR10 (top) and CIFAR100 (bottom).

15.4 Appendix

15.4.1 Datasets and Hyperparameters used in MCL Experiments

This section concludes by, in the form of tables and plots, providing more information about our experiments and experimental results for the algorithms mentioned above.

Dataset	News20	MNIST	CIFAR10	STL10	SVHN	Fashion
#Training	11314	50000	50000	5000	73257	50000
#Test	7532	10000	10000	8000	26032	10000
#Feature	129791	28×28	$32 \times 32 \times 3$	$96 \times 96 \times 3$	$32 \times 32 \times 3$	28×28
#Class	20	10	10	10	10	10

Table 15.5: Details regarding the datasets in MCL experiments.

Dataset	News20	MNIST	CIFAR10	STL10	SVHN	Fashion
p	50	50	20	20	20	50
#cluster	200	1000	1000	400	800	1000
γ	0.05	0.05	0.05	0.2	0.2	0.05
initial k	4	4	4	20	30	10
initial λ	6×10^{-6}	1×10^{-6}	8×10^{-7}	1×10^{-7}	1×10^{-6}	1×10^{-6}
initial η	3.5	0.02	0.01	0.02	0.01	0.02

Table 15.6: Parameters of MCL (Algorithm 2) and its variants for different datasets.

15.4.2 Additional Experiments of MCL

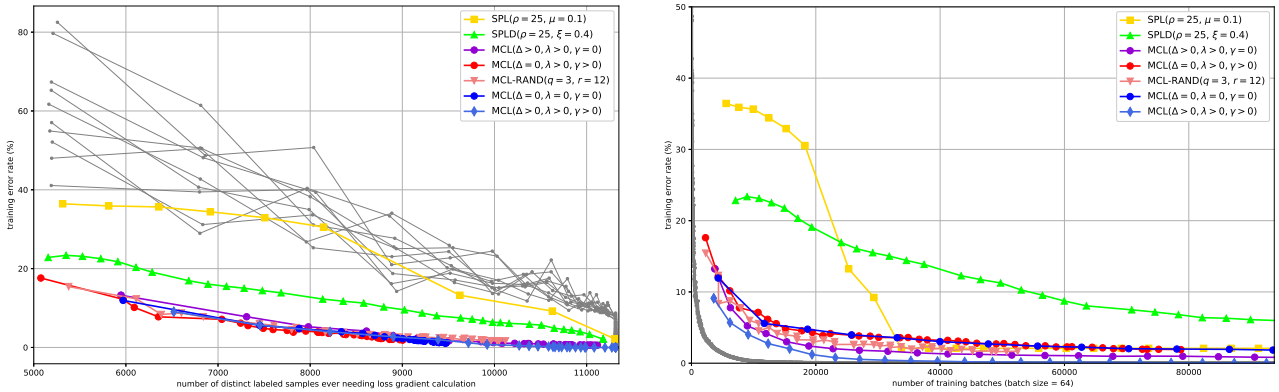

Figure 15.16: Training error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD).

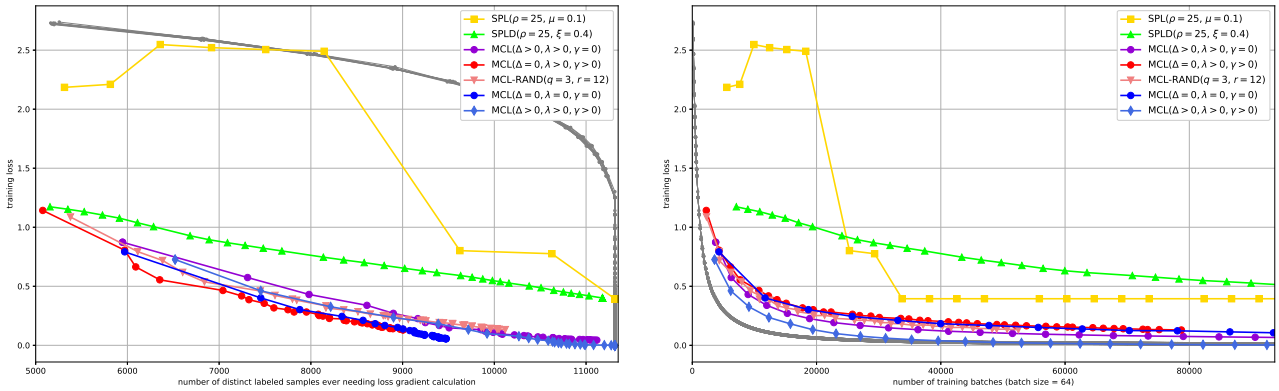

Figure 15.17: Training loss vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD).

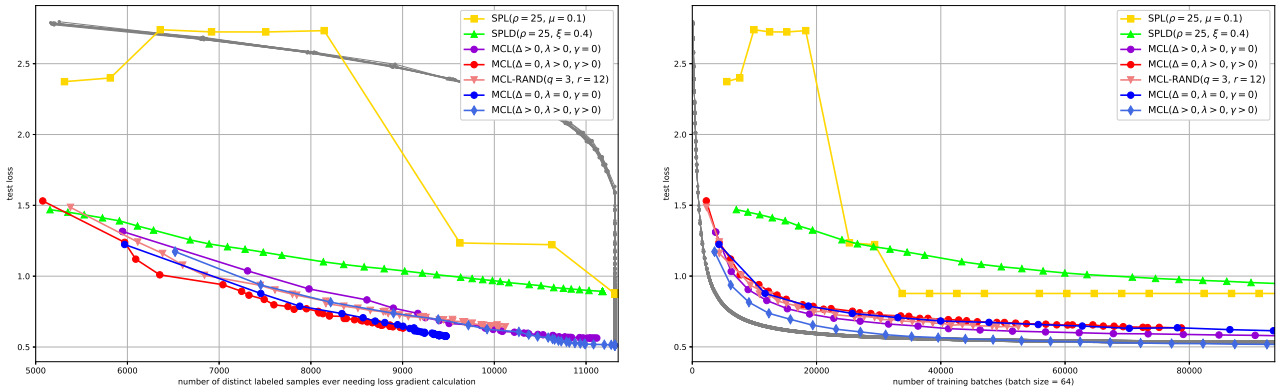

Figure 15.18: Test loss vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD).

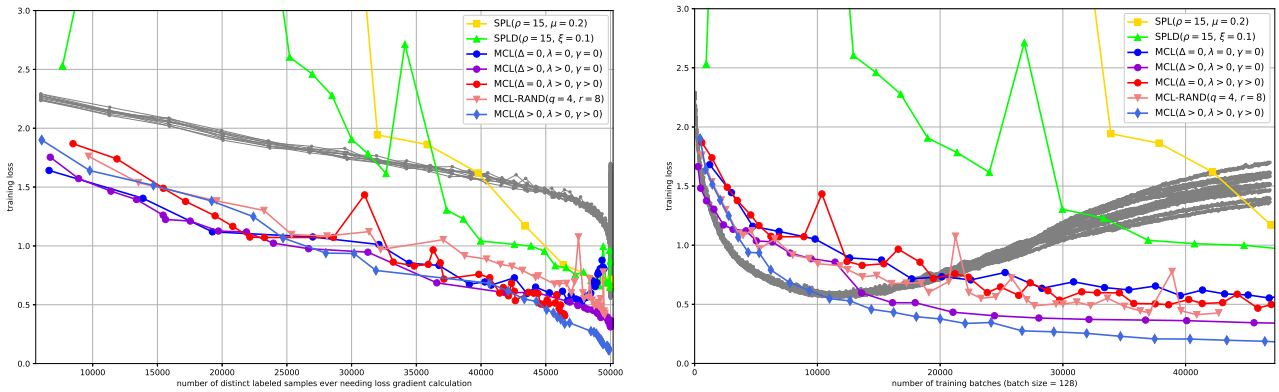

Figure 15.19: Training loss vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on CIFAR10 (grey curves represents 10 random trials of SGD).

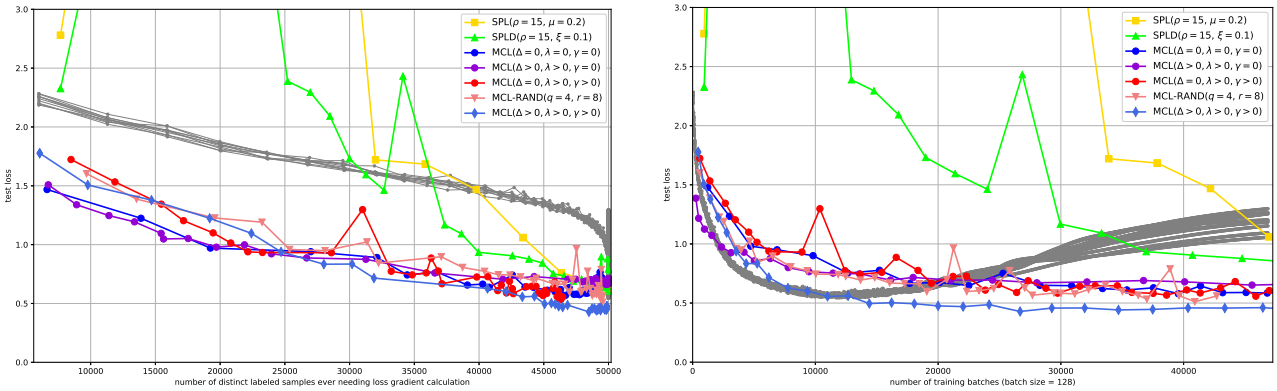

Figure 15.20: Test loss vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on CIFAR10 (grey curves represents 10 random trials of SGD).

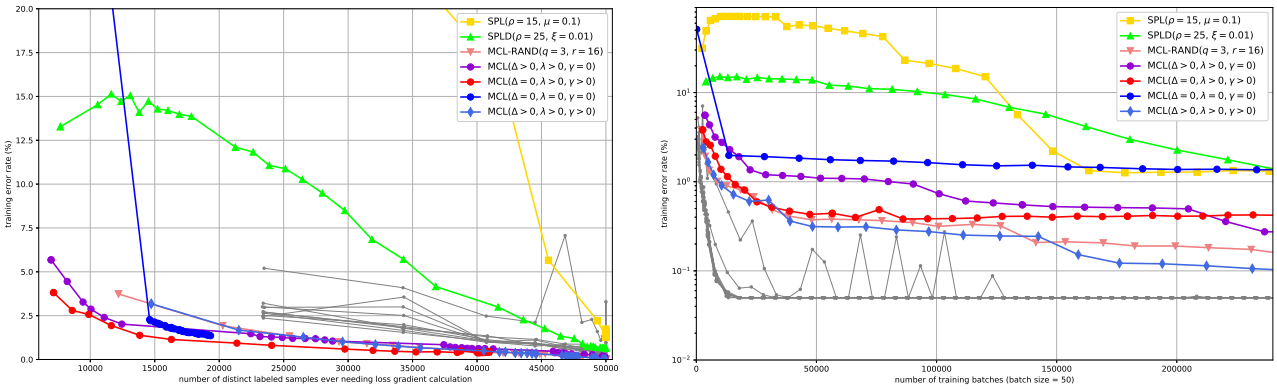

Figure 15.21: Training error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on MNIST (grey curves represents 10 random trials of SGD).

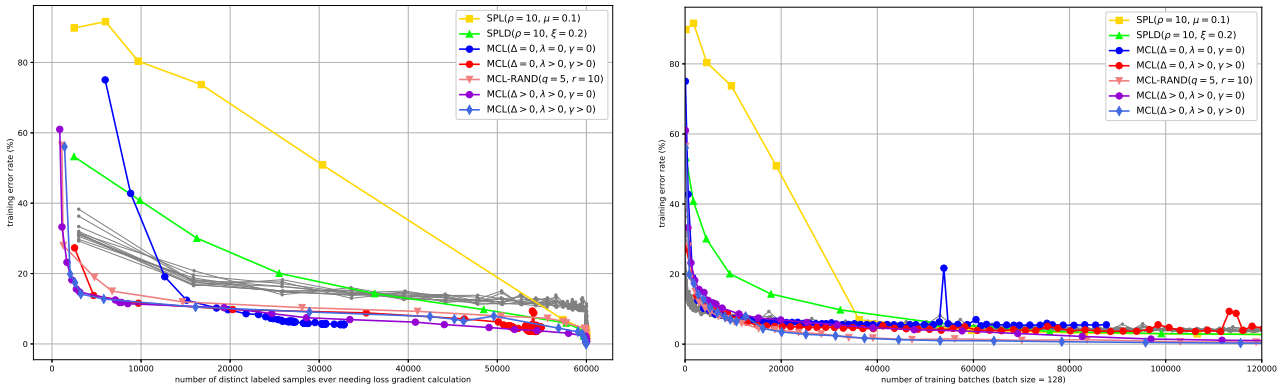

Figure 15.22: Training error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on Fashion-MNIST (grey curves represents 10 random trials of SGD).

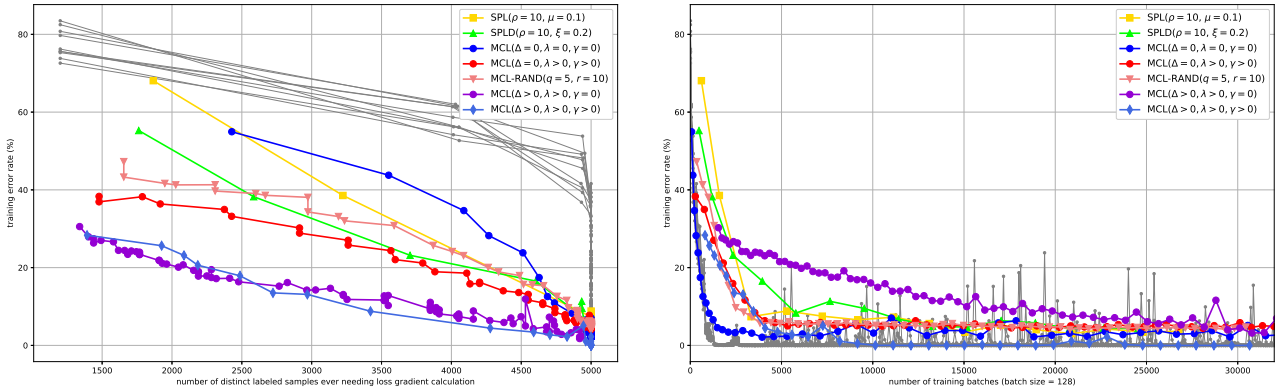

Figure 15.23: Training error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on STL10 (grey curves represents 10 random trials of SGD).

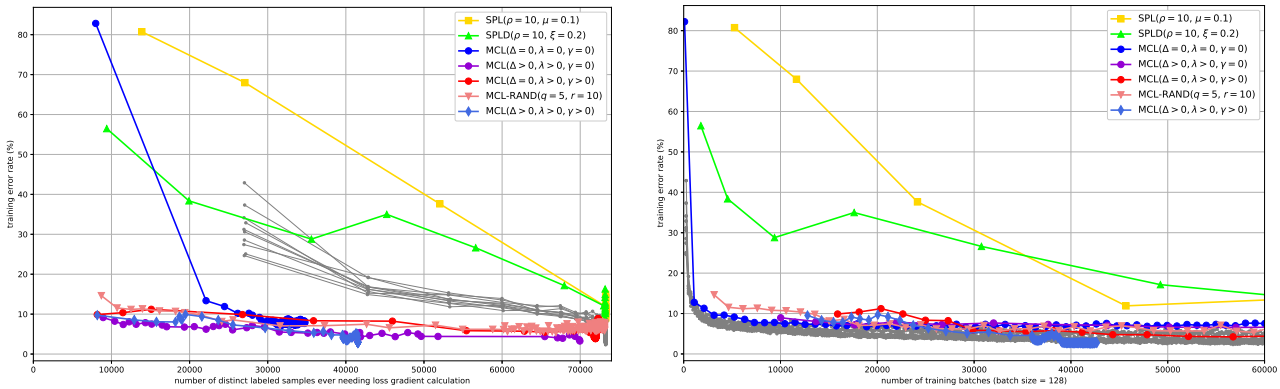

Figure 15.24: Training error rate (%) vs. number of distinct labeled samples ever needing loss gradient calculation (left) and number of training batches (right) on SVHN (grey curves represents 10 random trials of SGD).

15.4.3 Datasets and Hyperparameters used in DIHCL Experiments

Table 15.7: Details regarding the datasets and training settings (#Feature denotes the number of features after cropping if applied), "lr_start" and "lr_target" denote the starting and target learning rate for the first episode of cosine annealing schedule, they are gradually decayed over the remaining epochs.

Dataset	CIFAR10	CIFAR100	Food-101	ImageNet	STL10	SVHN
#Training	50000	50000	75750	1281167	5000	73257
#Test	10000	10000	25250	50000	8000	26032
#Feature	$(3, 32, 32)$	$(3, 32, 32)$	$(3, 224, 224)$	$(3, 224, 224)$	$(3, 96, 96)$	$(3, 32, 32)$
#Class	10	100	101	1000	10	10
#Epoch T	300	300	400	200	1200	300
BatchSize	128	128	80	256	128	128
lr_start	2×10^{-1}	2×10^{-1}	2×10^{-1}	2×10^{-1}	2×10^{-1}	2×10^{-2}
lr_target	5×10^{-4}	5×10^{-4}	1×10^{-4}	1×10^{-4}	5×10^{-4}	1×10^{-3}

We use cosine annealing learning rate schedule for multiple epochs. The switching epoch between each two consecutive episode for different datasets are listed below.

- CIFAR10, CIFAR100, SVHN, KMNIST, FMNIST:

Table 15.8: Details regarding the datasets and training settings (cont.)

Dataset	Birdsnap	FGVCaircraft	StanfordCARs	KMNIST	FMNIST
#Training	47386	6667	8144	50000	50000
#Test	2443	3333	8041	10000	10000
#Feature	$(3, 224, 224)$	$(3, 224, 224)$	$(3, 224, 224)$	$(1, 28, 28)$	$(1, 28, 28)$
#Class	500	100	196	10	10
#Epoch T	400	400	400	300	300
BatchSize	258	256	256	128	128
lr_start	4×10^{-1}	4×10^{-1}	4×10^{-1}	4×10^{-2}	4×10^{-2}
lr_target	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-3}	1×10^{-3}

$(5, 10, 15, 20, 30, 40, 60, 90, 140, 210, 300)$;

- STL10: $(20, 40, 60, 80, 120, 160, 240, 360, 560, 840, 1200)$
  $= 4 \times (5, 10, 15, 20, 30, 40, 60, 90, 140, 210, 300)$;
- ImageNet: $(5, 10, 15, 20, 30, 45, 75, 120, 200)$;
- Food-101, Birdsnap, FGVC-Aircraft, StanfordCars:
  $(10, 20, 30, 40, 60, 90, 150, 240, 400) = 2 \times (5, 10, 15, 20, 30, 45, 75, 120, 200)$;

15.4.4 Additional Experiments of DIHCL

We report how the test accuracy changes with the number of training batches for each method, and the wall-clock time for all the 11 datasets in Figure 15.25-15.28.

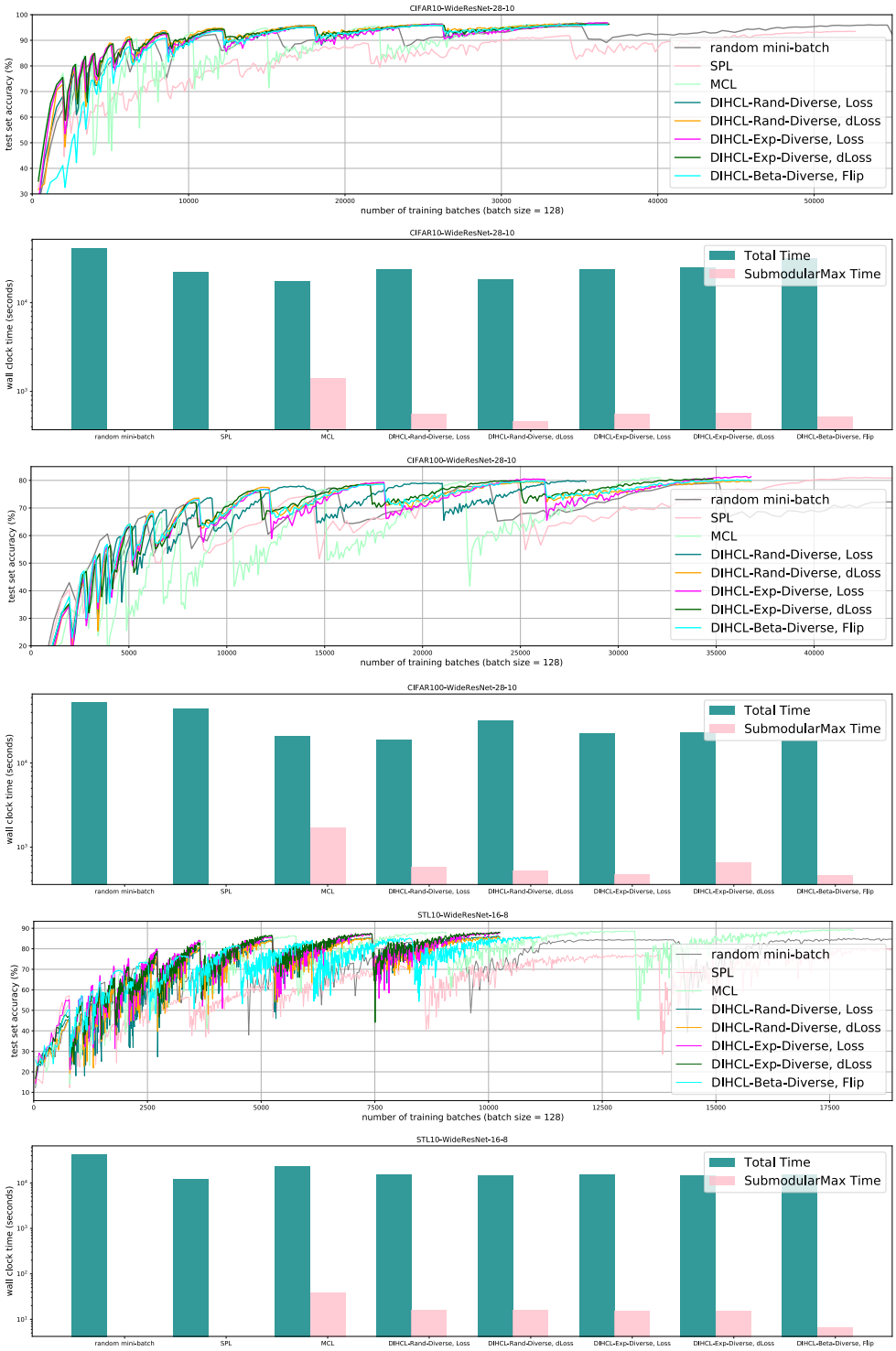

Figure 15.25: Training DNNs by using DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch SGD on 3 datasets, i.e., CIFAR10, CIFAR100 and STL-10. We use "Diverse" to denote DIHCL that further reduces S_t by applying submodular maximization for Eq. (10.2). We report how the test accuracy changes with the number of training batches for each method, and the (**log-scale**) wall-clock time for 1) the entire training and 2) the submodular maximization part in DIHCL with diversity and MCL.

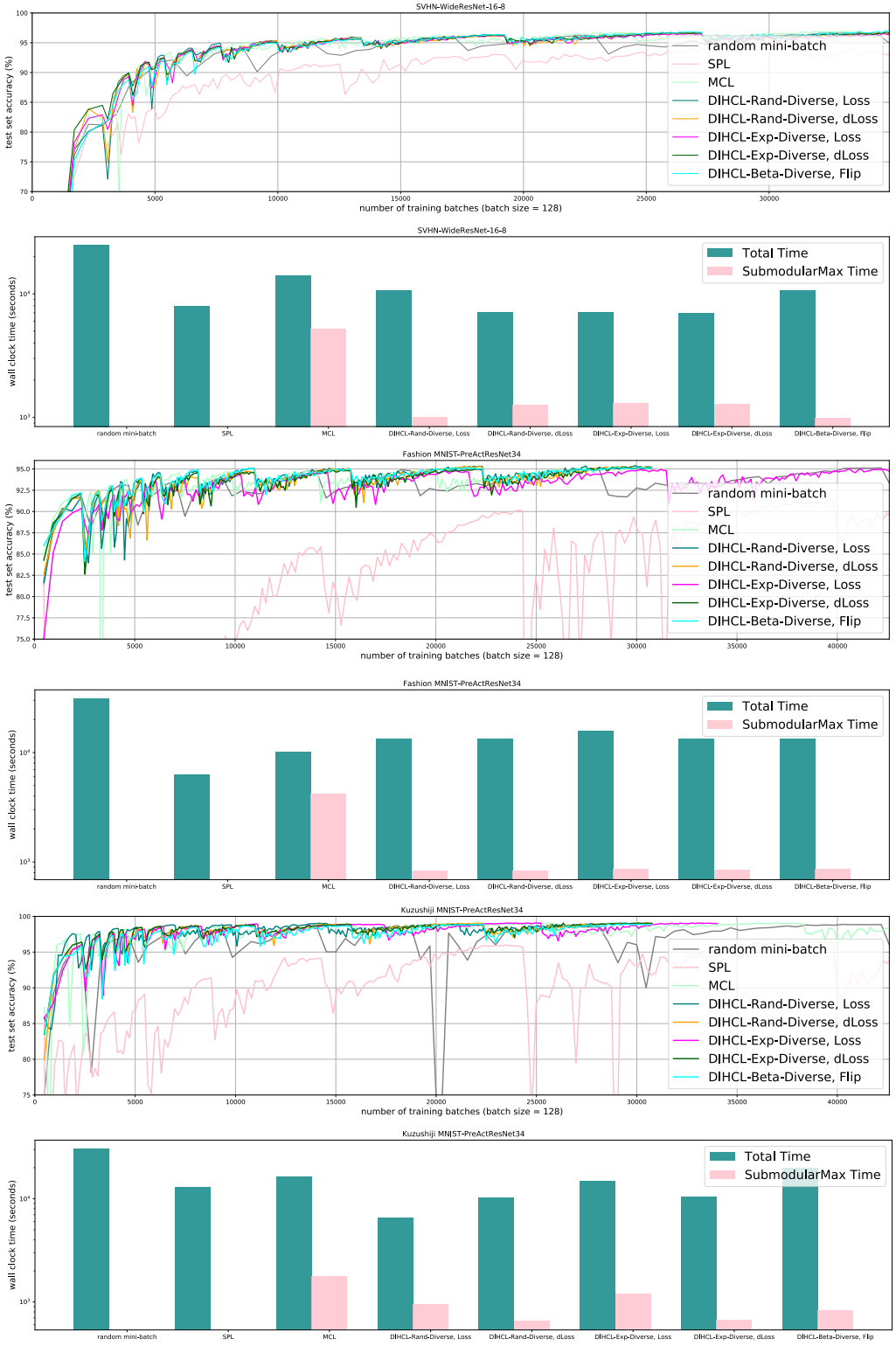

Figure 15.26: Training DNNs by using DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch SGD on 3 datasets, i.e., SVHN, Fashion MNIST and Kuzushiji MNIST. We use "Diverse" to denote DIHCL that further reduces S_t by applying submodular maximization for Eq. (10.2). We report how the test accuracy changes with the number of training batches for each method, and the (**log-scale**) wall-clock time for 1) the entire training and 2) the submodular maximization part in DIHCL with diversity and MCL.

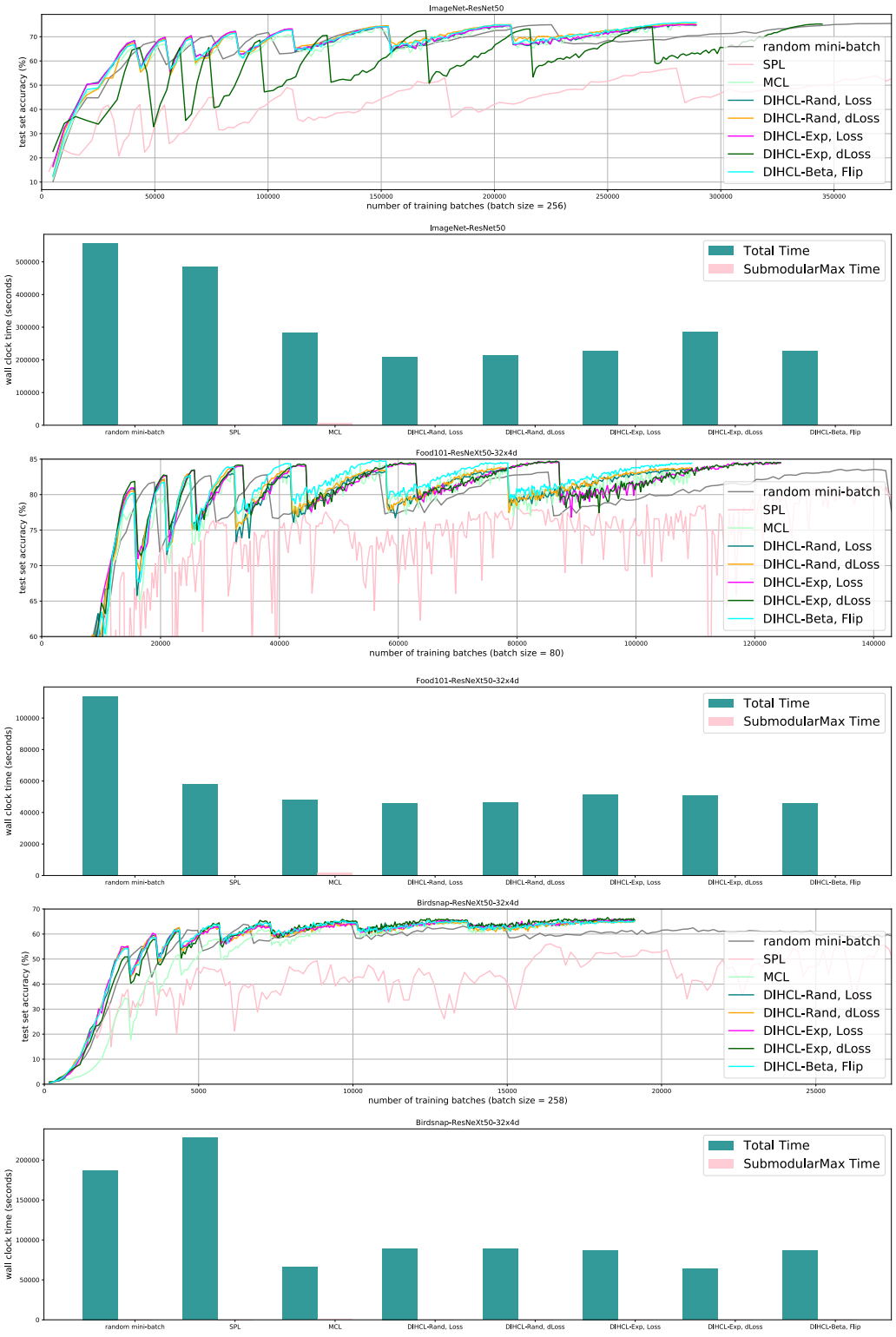

Figure 15.27: Training DNNs by using DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch SGD on 3 datasets, i.e., ImageNet, Food-101 and Birdsnap. We report how the test accuracy changes with the number of training batches for each method, and the wall-clock time for 1) the entire training and 2) the submodular maximization part in MCL.

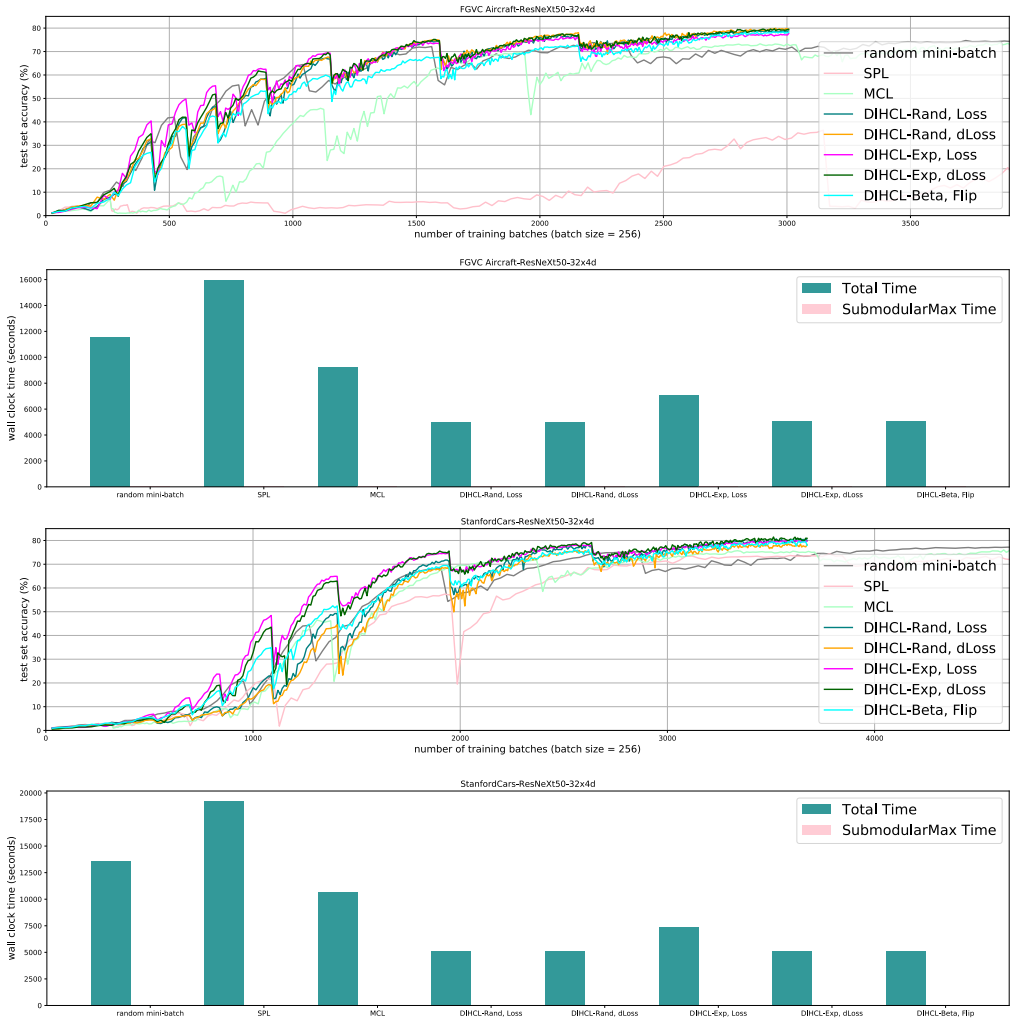

Figure 15.28: Training DNNs by using DIHCL (and its variants), SPL [116], MCL [248], and random mini-batch SGD on 2 datasets, i.e., FGVC Aircraft and Stanford Cars. We report how the test accuracy changes with the number of training batches for each method, and the wall-clock time for 1) the entire training and 2) the submodular maximization part in MCL.

15.4.5 Datasets and Hyperparameters used in DoCL Experiments

On each dataset, for all the methods, we use the same cosine annealing learning rate schedule for multiple episodes. The ending epochs of cycles $\{T_i\}_{i=0}^{\kappa}$ in our learning rate schedule used on different datasets are listed below.

- CIFAR10, CIFAR100, SVHN, FMNIST: $(5, 10, 15, 20, 30, 40, 60, 90, 140, 210, 300)$;

- ImageNet: $(5, 10, 15, 20, 30, 45, 75, 120, 200)$;
- Food-101, Birdsnap, FGVCaircraft, StanfordCars (double the ImageNet cycles):

  $(10, 20, 30, 40, 60, 90, 150, 240, 400) = 2 \times (5, 10, 15, 20, 30, 45, 75, 120, 200)$;

Table 15.9: Details regarding the datasets and training settings (#Feature denotes the number of features after cropping if applied), "lr_start" and "lr_target" denote the starting and target learning rate for the first episode of cosine annealing schedule, they are gradually decayed over the rest episodes.

Dataset	CIFAR10	CIFAR100	Food-101	ImageNet	SVHN
#Training	50000	50000	75750	1281167	73257
#Test	10000	10000	25250	50000	26032
#Feature	$(3, 32, 32)$	$(3, 32, 32)$	$(3, 224, 224)$	$(3, 224, 224)$	$(3, 32, 32)$
#Class	10	100	101	1000	10
#Epoch T	300	300	400	200	300
BatchSize	128	128	80	256	128
lr_start	2×10^{-1}	2×10^{-1}	2×10^{-1}	2×10^{-1}	2×10^{-2}
lr_target	5×10^{-4}	5×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-3}
weight decay	1×10^{-4}	1×10^{-4}	1×10^{-5}	1×10^{-5}	1×10^{-4}

Dataset	Birdsnap	FGVCaircraft	StanfordCARs	FMNIST
#Training	47386	6667	8144	50000
#Test	2443	3333	8041	10000
#Feature	$(3, 224, 224)$	$(3, 224, 224)$	$(3, 224, 224)$	$(1, 28, 28)$
#Class	500	100	196	10
#Epoch T	400	400	400	300
BatchSize	258	256	256	128
lr_start	4×10^{-1}	4×10^{-1}	4×10^{-1}	4×10^{-2}
lr_target	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-3}
weight decay	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-4}

Chapter 16

WEAKLY-SUPERVISED LEARNING

In this chapter, we mainly focus on weakly-supervised learning and evaluate two curriculum learning algorithms in the experiments, i.e., time-consistent semi-supervised learning (TC-SSL) proposed in Chapter 12 and robust curriculum learning (RoCL) proposed in Chapter 13.

16.1 Time-Consistency Curriculum for Semi-Supervised Learning

Semi-supervised learning (SSL) leverages unlabeled data when training a model with insufficient labeled data. A common strategy for SSL is to enforce the consistency of model outputs between similar samples, e.g., neighbors or data augmentations of the same sample. However, model outputs can vary dramatically on unlabeled data over different training stages, e.g., when using large learning rates. This can introduce harmful noises and inconsistent objectives over time that may lead to concept drift and catastrophic forgetting.

In Section 8.3, we study the dynamics of neural net outputs in SSL and discover that "*time-consistency* (TC)" score is able to identify the samples with stable model predictions and accurate pseudo-labels. In Chapter 12, we develop an efficient SSL algorithm based on the TC score, i.e., "time-consistent semi-supervised learning (TC-SSL)". In this section, by extensive experiments, we show that TC-SSL selecting and using first the unlabeled samples with more consistent outputs over the course of training can improve the final test accuracy and save computation. Moreover, TC-SSL achieves the best test-set accuracy on several SSL benchmarks and significantly reduces the computational cost required by previous methods.

16.1.1 Semi-Supervised Learning

Semi-supervised learning (SSL) tackles a challenging problem commonly encountered in real-world applications, i.e., how one should train a reliable machine learning model with limited amounts of labeled samples but a rich reserve of unlabeled samples. For many tasks, collecting accurate labeled data is expensive and error-prone as it requires human labor, while unlabeled data is cheap and abundant. Commonly used methodology in the early days of SSL includes label and measure propagation [260, 247, 200] and manifold regularization [163], which encourage the label consistency within a local neighborhood on the embedded data manifold (i.e., the model should produce similar label distributions for samples geodesically close to each other), and where the manifold is approximated by a graph. This encourages the model to be locally smooth on the manifold.

Recent deep learning shows that a trained neural net not only works as an accurate classifier but also provides hidden-layer features that can capture manifold structure at multiple scales [221, 241] Moreover, with prior knowledge of data and inductive bias embedded in the neural net architecture and training strategies, e.g., convolutions and data augmentations [243, 52, 47, 45], we can train a more locally consistent model with better generalization performance. This motivates recent progress on self-supervision [173, 214, 245, 164] and SSL [181, 90, 155, 8, 118, 22, 165], in which a self-supervised loss (which does not require ground-truth labels) defined on data augmentations can encourage a neural network's local smoothness on the data manifold. In Section 12.1, we have discussed two types of self-supervised loss, i.e., "consistency regularization" [176] and "contrastive loss" [38] or a "triplet loss" [232]. The former encourages that a sample and its augmentations should have similar outputs/representations, while the latter enforces different samples (and their augmentations) having different outputs/representations. Neural networks trained by this strategy have achieved substantial improvements on SSL [181, 21] and unsupervised learning [32, 80] tasks without leveraging the weighed neighborhood graphs commonly used in

earlier works. More recently, MixMatch [22] and ReMixMatch [21] utilize a consistency loss with data augmentations and, with a delicate combination of regularization techniques, gets significantly improved SSL performance on several datasets, thus demonstrating the effectiveness of combining data augmentation and self-supervision.

The self-supervised loss defined on data augmentations can be explained as an ordinary supervised loss with a pseudo target [122, 155] generated by the neural net itself or a self-ensemble [209, 118]. For a consistency loss, the pseudo target for an unlabeled sample can be the class distribution of the sample's random augmentation(s) generated by the neural net. For a contrastive loss, we can use softmax to normalize the similarities between a sample and a subset of other samples (including one of its own augmentations). The pseudo target for the softmax output has value 1 for the sample's augmentation and 0 for the other samples [216]. Although pseudo targets achieved on data augmentations tend to be accurate, the above techniques cannot guarantee time-consistent pseudo targets for unlabeled samples. At different training stages, the output of a neural net for a sample can change dramatically due to the non-smoothness of activation functions, the complexity of network structures, and sophisticated learning rate schedules [193, 142, 132]. In such a case, when we use self-supervised losses with varying pseudo targets for the same sample, the optimization objectives may keep changing over training steps. This inconsistency of the objective over time can considerably slow down the training progress or even make it diverge, resulting in serious concept drift and training failure.

Another common strategy for SSL is to select samples whose predictions have high confidence. However, the confidence can still change drastically over time for the same aforementioned reason. Moreover, a neural network can often simultaneously be overconfident and incorrect on training samples. Indeed, our later empirical analysis shows that confidence is an unreliable metric for selecting samples. In addition, samples with high confidence tend to be less informative to the future training or slow down the training since the gradient is proportional to the gap between current

outputs and the pseudo target, which is small if the confidence is high.

We then present an empirical study of TC for unlabeled data selection and compare this with using confidence. We artificially split a fully labeled dataset into labeled and unlabeled subsets, and train a neural net by only using the labeled set. We observe that the model tends to produce consistent outputs on some unlabeled samples but frequently changes its prediction on others. We further find that the time-consistent outputs (the former) are usually correct predictions while the latter contains more mistakes. The observations lead us to a natural curriculum [19, 248] for SSL that selects unlabeled samples with higher TC at each training step while gradually increasing the selection budget. We then introduce "TC-SSL" that adopts the curriculum to train a neural net by minimizing the consistency loss and contrastive loss defined on augmentations of unlabeled samples (using learned policies of AutoAugment [47]). In experiments, we show that TC-SSL outperforms the very recent MixMatch and other SSL approaches on three datasets (CIFAR10, CIFAR100, and STL10) under various labeled-unlabeled splittings and significantly improves SSL efficiency, i.e., consistently using $< 20\%$ training batches of what the best baseline needs. These results portend well for TC-SSL as a modern SSL methodology.

16.1.2 Related Work of Semi-Supervised Learning

Classic SSL methods enforce samples to have similar label distributions in every local region of the data manifold — this approach is taken by label/measure propagation [260, 261, 247, 200, 18, 90] and manifold regularization [163, 15]. They rely on similarity measures between samples to find the local neighborhood for every sample. One can think of these methods as encouragements of "spatial consistency". It is not always clear, however, how to determine sample pair similarity for a given data set and model. E.g., which feature space do we compute the similarity in? What similarity metric performs best? Recently, graph neural net (GNN) based methods [108, 220, 222] can apply graph-based SSL on neural net architectures and gets compelling performance on SSL of graph data.

The data in these works consists of natural graph structures, while on arbitrary data, generating the graph is itself a challenging problem.

Recent SSL research encourages spatial consistency by combining data augmentation techniques and self-supervised losses. In [176, 181], the authors propose "consistency regularization" to encourage the sample and its augmentations to have similar neural net outputs. "Contrastive loss" [38, 216] and "triplet loss" [232, 187], on the contrary, encourage different samples (and their augmentations) to have distant outputs. MixMatch [22] combines several techniques, i.e., consistency loss + sharpening the pseudo target distribution averaged over multiple augmentations + MixUp [243], which significantly improve SSL performance on several datasets. Compared to graph-based SSL, it integrates multiple augmentation methods [21, 47, 45] to provide stronger self-supervision. Our work also adopts a similar strategy but additionally enforces the time-consistency of selected samples at every SSL step, which is an orthogonal technique that can be seamlessly integrated into MixMatch.

16.1.3 Experiments

Table 16.1: Test error rate (mean±variance) of SSL methods training a small WideResNet and a large WideResNet on **CIFAR10**. Baselines: Pseudo Label [122], Π-model [181], VAT [155], Mean Teacher [209], MixMatch [22], ReMixMatch [21].

Benchmark	CIFAR10 (small WideResNet-28-2)				CIFAR10 (large WideResNet-28-135)			
labeled/unlabeled	500/44500	1000/44000	2000/43000	4000/41000	500/44500	1000/44000	2000/43000	4000/41000
Pseudo Label	40.55 ± 1.70	30.91 ± 1.73	21.96 ± 0.42	16.21 ± 0.11	-	-	-	-
Π-model	41.82 ± 1.52	31.53 ± 0.98	23.07 ± 0.66	5.70 ± 0.13	-	-	-	-
VAT	26.11 ± 1.52	18.68 ± 0.40	14.40 ± 0.15	11.05 ± 0.31	-	-	-	-
Mean Teacher	42.01 ± 5.86	17.32 ± 4.00	12.17 ± 0.22	10.36 ± 0.25	-	-	-	-
MixMatch	9.65 ± 0.94	7.75 ± 0.32	7.03 ± 0.15	6.24 ± 0.06	8.44 ± 1.04	7.38 ± 0.63	6.51 ± 0.48	5.12 ± 0.31
ReMixMatch	-	$\mathbf{5.73 \pm 0.16}$	-	$5.14 \pm \mathbf{0.04}$	-	-	-	-
TC-SSL (ours)	$\mathbf{9.14 \pm 0.88}$	6.15 ± 0.23	$\mathbf{5.85 \pm 0.10}$	$\mathbf{5.07} \pm 0.05$	$\mathbf{6.04 \pm 0.39}$	$\mathbf{3.81 \pm 0.19}$	$\mathbf{3.79 \pm 0.21}$	$\mathbf{3.54 \pm 0.06}$

In this section, we apply TC-SSL to train Wide ResNet [240] models on different labeled-unlabeled splittings of three datasets, i.e., CIFAR10, CIFAR100 [114], and STL10 [40]. Unless

otherwise specified, we followed all the settings and train the same neural net architectures from previous works such as MixMatch [22] and mean teacher [209] to make sure that the comparison with previously published results on these datasets are fair. For CIFAR10 experiments, we train a small WideResNet-28-2 (28 layers, width factor of 2, 1.5-million parameters) and a large WideResNet-28-135 (28 layers, 135 filters per layer, 26-million parameters) for four kinds of labeled/unlabeled/validation random splittings applied to the original training set of CIFAR10, i.e., 500/44500/5000, 1000/44000/5000, 2000/43000/5000, 4000/41000/5000. For CIFAR100 experiments, we train WideResNet-28-135 for three splittings, i.e., 2500/42500/5000, 5000/40000/5000, 10000/35000/5000. For STL10 experiment, we train a deeper variant of WideResNet-28-2 with four extra residual blocks added before the output layer with the purpose of processing larger images in STL10 (compared with CIFAR images). This extra part as a whole has 256 input channels and 512 output channels and increases the total parameters from 1.5-million to 5.9-million, which however is still much smaller than the model used in MixMatch (23.8-million parameters) [22]. We randomly draw 500 samples from the original training set as our validation set, so the splitting is 4500/100000/500. We evaluate the performance of all the trained models on the original test set of each dataset.

We run Pytorch re-implementation of MixMatch* to achieve MixMatch's results of training large WideResNet-28-135 on CIFAR10 and CIFAR100. All the other baselines' results are from MixMatch and ReMixMatch paper [22, 21]. Due to limited computation resources, we cannot run experiments for all the baselines and any missing results are marked by "-". We run our Pytorch implementation of TC-SSL for each experiment on one NVIDIA GeForce RTX2080Ti or TESLA V100.

Table 16.2: Test error rate (mean±variance) of SSL methods training WideResNet-28-135 on **CIFAR100**. Baselines: SWA [8], MixMatch [22].

Benchmark	CIFAR100 (WideResNet-28-135)		
labeled/unlabeled	2500/42500	5000/40000	10000/35000
SWA	-	-	28.80
MixMatch	44.20 ± 1.18	34.62 ± 0.63	$25.88 \pm \mathbf{0.30}$
TC-SSL (ours)	$\mathbf{31.95 \pm 0.55}$	$\mathbf{26.98 \pm 0.51}$	$\mathbf{22.10} \pm 0.37$

Table 16.3: **Ablation Study:** test error rate (mean±variance) of TC-SSL variants for training WideResNet-28-135 on CIFAR10: "no consistency" removes the consistency loss in Eq. (12.2); "no contrastive" removes the contrastive loss in Eq. (12.3); "no PseudoLabel" removes the cross entropy loss for unlabeled data in Eq. (12.5); "no TC-selection" replaces Line 8-9 of Algorithm 7 with uniform sampling.

labeled/unlabeled	500/44500	1000/44000	2000/43000	4000/41000
TC-SSL (ours)	$\mathbf{6.04 \pm 0.39}$	$\mathbf{3.81 \pm 0.19}$	$\mathbf{3.79 \pm 0.21}$	$\mathbf{3.54 \pm 0.06}$
TC-SSL (no consistency)	7.51 ± 0.56	5.31 ± 0.23	3.82 ± 0.20	$3.58 \pm \mathbf{0.06}$
TC-SSL (no contrastive)	7.56 ± 0.52	5.35 ± 0.28	3.96 ± 0.25	3.66 ± 0.08
TC-SSL (no PseudoLabel)	41.05 ± 2.32	23.64 ± 1.17	14.37 ± 0.69	9.87 ± 0.22
TC-SSL (no TC-selection)	12.25 ± 0.81	6.39 ± 0.44	4.68 ± 0.35	4.05 ± 0.13

Hyperparameters

For TC-SSL in the experiments, we apply $T_0 = 10$ warm starting epochs and $T = 680$ epochs in total. Note the "epoch" here refers to one iteration in Algorithm 7 and is different from its meaning in most fully supervised training, where it refers to a full pass of the whole training set. In our case, the training samples in each epoch changes according to our curriculum of k^t. We apply SGD with momentum of 0.9 and weight decay of 2×10^{-5}, and use a modified cosine annealing learning rate schedule [142] for multiple episodes of increasing length and decaying target learning rate, since it can quickly jump between different local minima on the loss landscape and explore more regions without being trapped by a bad local minima. In particular, we set up 12 episodes with epochs-per-episode starting from 10 (i.e., the warm starting episode) and increasing by 10

[*] https://github.com/YU1ut/MixMatch-pytorch

after every episode until reaching epoch-680. The learning rate at the beginning and end of the first episode are set to 0.2 and 0.02, respectively. We then multiply each of them by 0.9 after every episode. We do not heavily tune the λ-parameters and γ-parameters. For all experiments, we use $\lambda_{cs} = {}^{20}/_{C}, \lambda_{ct} = 0.2, \lambda_{ce} = 1.0, \gamma_\theta = \gamma_c = 0.99, \gamma_k = 0.005$ (C is the number of classes). For data augmentation, we use AutoAugment [47] learned policies for the three datasets followed by MixUp with the mixing weight sampled from Beta$(0.5, 0.5)$. We initialize $k^1 = 0.1|\mathcal{U}|$ and θ^0 by Pytorch default initialization. We apply all the practical tips detailed in Section 12.3.

We do not heavily tune the hyperparameters for computational reasons. We tested limited options for $\gamma_\theta, \gamma_c, \lambda_{ct}, \lambda_{cs}$ and chose the ones with greatest improvement on validation set after 5 episodes. We only tune them on CIFAR10 with 500 labeled samples, and use the same hyperparameters for all other experiments.

The hyperparameters in TC-SSL can be divided into three groups: epochs T_0 and T, regularization weights λs, and multiplicative factors γs:

(1) For epochs, we fix warm starting epochs as $T_0 = 10$ since 10 epochs of training usually results in a reasonable model to start with; we limit the total epochs $T = 680$ due to limited computation.

(2) For γs, we fix $gamma_k = 0.005$ so the last episode is trained on all unlabeled data, i.e., $k = |U|$. We tried several common discounting factors for $\gamma_\theta, \gamma_c = 0.995, 0.99, 0.98$.

(3) For λs, we fixed $\gamma_{ce} = 1$ so the selected unlabeled and labeled samples have equal weights of classification loss in the objective. In tuning, we tried $\lambda_{ct} = 0.2, 0.4, 0.8$ and $\lambda_{cs} = 10/C, 20/C, 40/C$.

Test Accuracy of TC-SSL

For each experiment, we run 5 trials with different random seeds and report the mean and variance of the test accuracy in Table 16.1-16.4). TC-SSL achieves the lowest test error on most experiments

Table 16.4: Test error rate of several methods training CNNs of different #params on **STL10**. Baselines: CutOut [52], IIC [97], MixMatch [22].

Benchmark	STL10
labeled/unlabeled	5k/100k
CutOut (11.0M)	12.74
IIC (N/A)	11.20
MixMatch (23.8M)	5.59
TC-SSL (ours, **5.9M**)	**4.82**

compared with several recent baselines using a heavy combination of multiple advanced techniques. Most of our results are SOTA, e.g., TC-SSL trains a much smaller model (5.9M parameters) and achieves the SOTA performance on STL10. Note these improvements are achieved in the case that we do not conduct any heavy tuning or fine-grained search of hyperparameters, nor use any advanced data augmentation. The only exception when TC-SSL performs worse happens at the 1000/44000 splitting of CIFAR10 when training a small WideResNet-28-2, in which case the very recent ReMixMatch achieves slightly better results than ours. However, ReMixMatch's result was achieved when using a powerful augmentation proposed in their paper, i.e., CTAugment, and averaging over multiple (i.e., 8 vs.1 in TC-SSL) augmentations to produce the pseudo targets. According to their ablation study [21], these two techniques bring significant improvements. In Table 16.3, we further present a thorough ablation study that separately verifies the improvement caused by each component of TC-SSL.

In addition, we provide a fine-grained comparison between MixMatch and TC-SSL on their test accuracy during the training process.

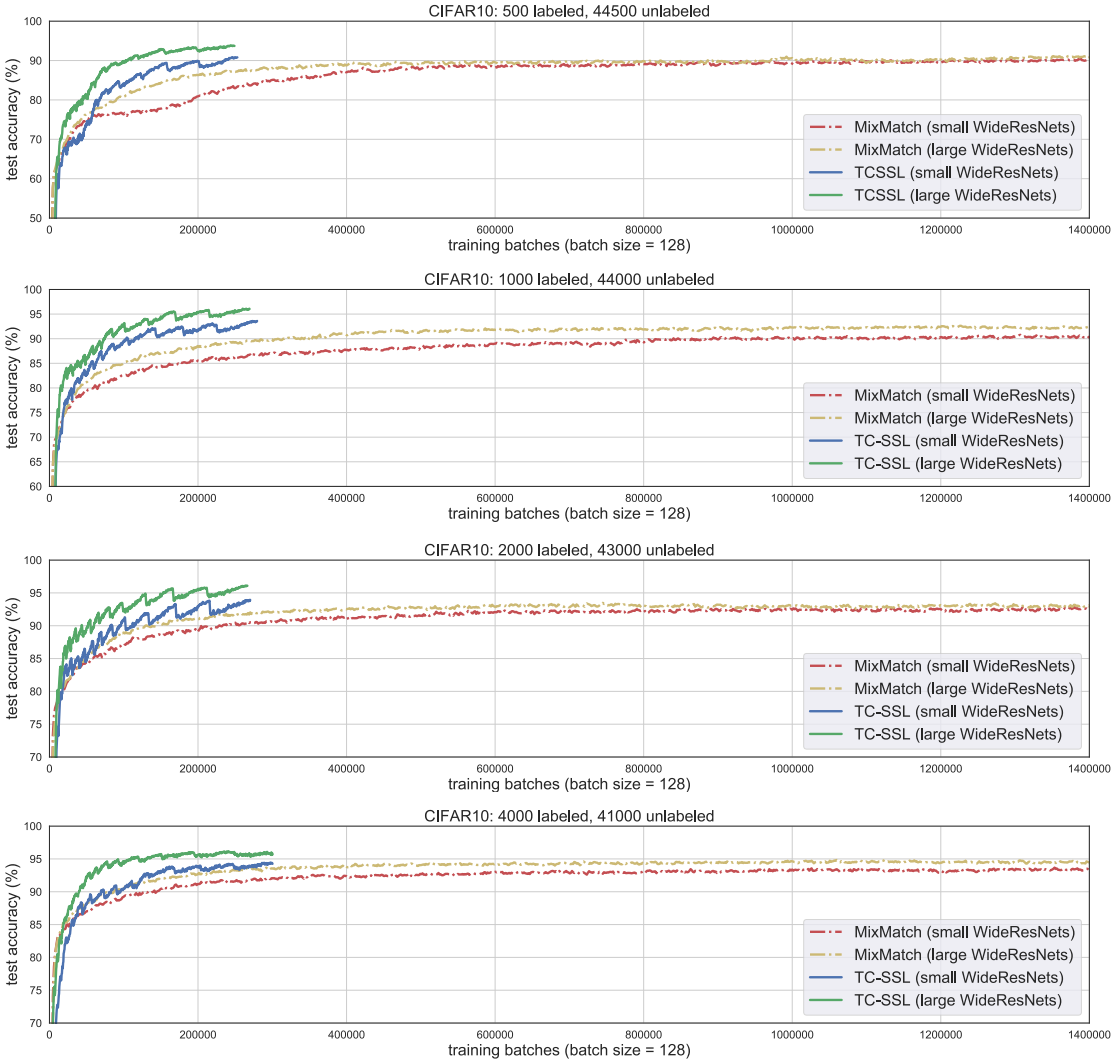

Figure 16.1: Test accuracy (%) during the training of small WideResNet and large WideResNet by MixMatch and TC-SSL on the four splittings of CIFAR10.

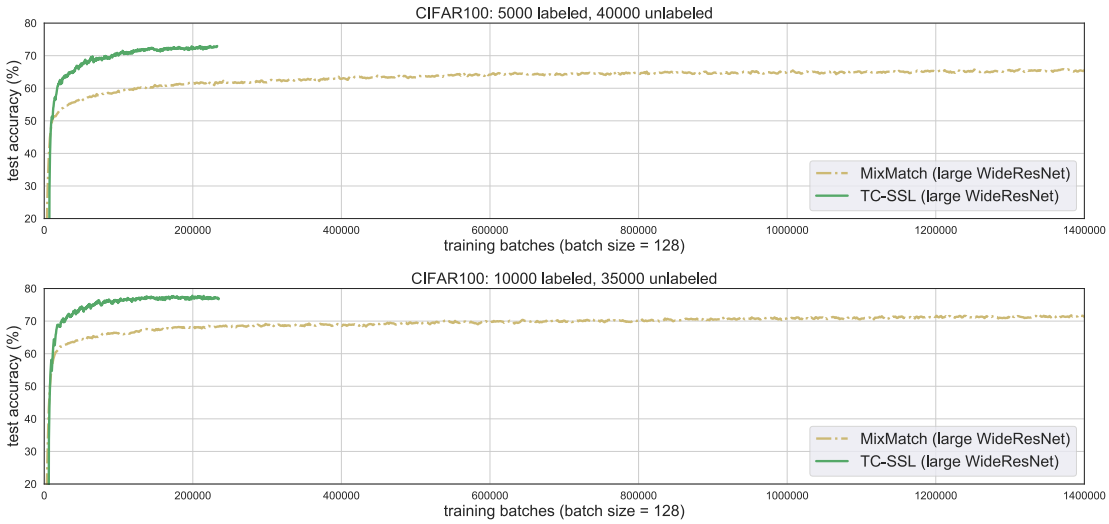

Figure 16.2: Test accuracy (%) during the training of the large WideResNet by MixMatch and TC-SSL on the two splittings of CIFAR100.

Training Efficiency of TC-SSL

Since the recent MixMatch achieves much better performance than the SSL methods prior to it, we present a more detailed comparison to it in Figure 16.1 by using the Pytorch re-implementation of MixMatch. We use the number of mini-batches for training in the two methods as an approximate metric of their training costs since the forward and back-propagation on them dominates the incurred computation. Since TC-SSL uses batch size of 128 while MixMatch uses 64, we divide the number of training batches in MixMatch by 2. The comparison shows TC-SSL's significant advantage in learning efficiency, e.g., it achieves higher test accuracy much earlier than MixMatch. This is because that TC-SSL selects the most time-consistent unlabeled samples that are more informative and less harmful to the training, where the selection is guided by an efficient curriculum. In contrast, MixMatch uses all the unlabeled samples since the beginning, which might introduce wrong training signals and noise, especially during earlier stages, and thus slow down the growth of test accuracy.

Quality of Pseudo Labels for Unlabeled Data Selected for Training in TC-SSL

One primary reason for the substantial improvements achieved by TC-SSL is that the unlabeled samples selected by time consistency have high-quality pseudo labels. To verify this, we report how the precision and recall of these selected pseudo labels together with their percentage in all the unlabeled data change during training in Figure 16.3.

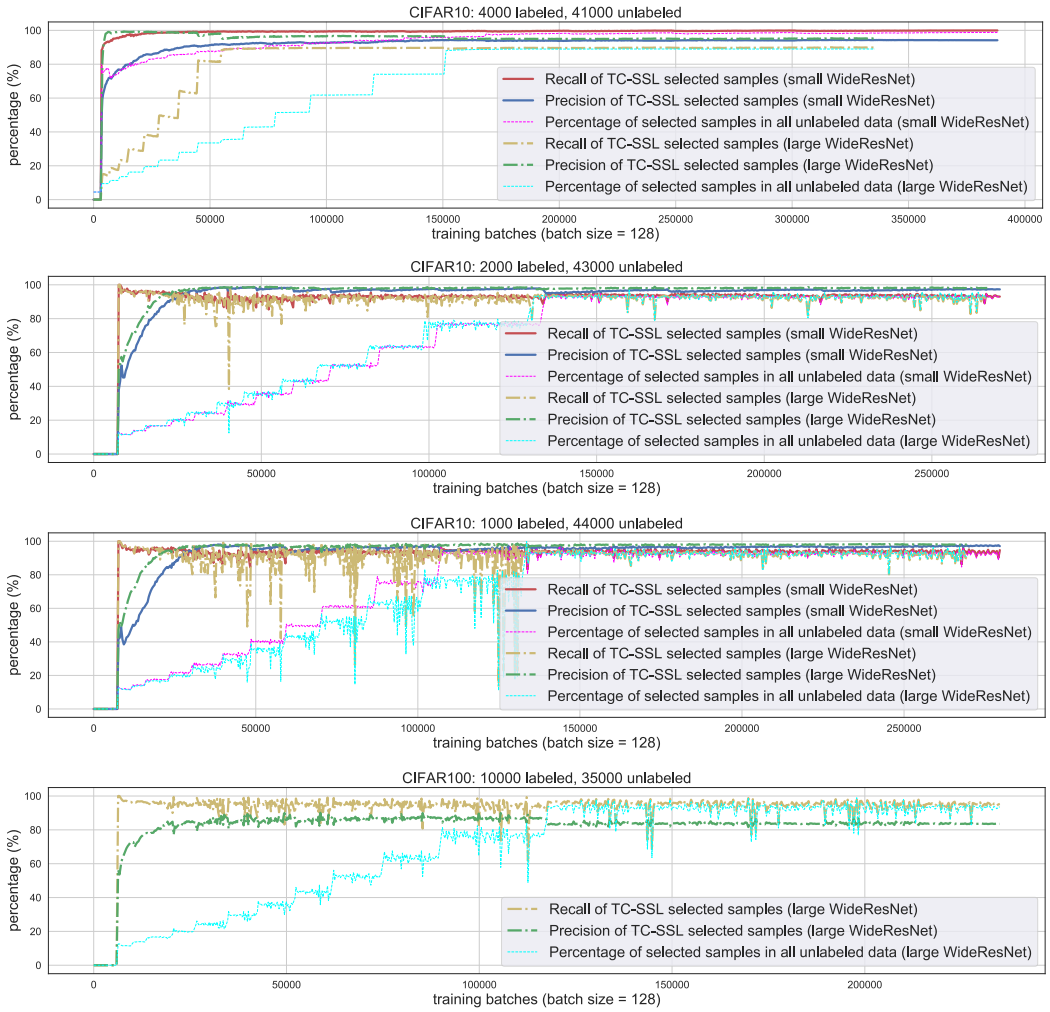

Figure 16.3: Precision and recall (%) of pseudo labels produced by the model during training for the unlabeled samples selected by TC-SSL. We also provide the percentage (%) of the selected samples in the ground set of unlabeled data $\mathcal{U}$.

In most of the four plots, the recall quickly increases to a high value close to 100% and keeps it high consistently over training, indicating that almost all the correctly-predicted unlabeled samples are selected by TC-SSL. In addition, the precision also increases quickly to a relatively high value (close to the test accuracy of fully-supervised learning that uses the whole training set), implying that most samples selected by TC-SSL have correct pseudo labels. This well explains the promising learning efficiency and final performance of TC-SSL. The noisy part (i.e., quick drops) on the recall curve when training large WideResNet is caused by the confidence thresholding mentioned in Section 12.3, which removes the extremely confident samples from S^t since they provide very limited information to training (though their pseudo labels might be correct). The curves are flat during the first few training batches since they are warm-starting epochs only using labeled samples and no unlabeled data is selected. It also worth noting that the model cannot be precise on predicting all the unlabeled data in the early stages, so selecting too many unlabeled data will leads to a large portion of incorrect pseudo labels used for training, i.e., low precision. In contrast, for TC-SSL, we adopt a curriculum of gradually increasing number of selected unlabeled data, which avoid the above problem and remains a high precision on the unlabeled data selected for early training steps.

Hence, TC-SSL can maintain a precision and recall close to 100% during most steps (except when it excludes some highly-confident and correctly predicted samples since they are less informative), while gradually increase the percentage of selected unlabeled samples to 100%.

16.2 Noisy-Label Learning by Robust Curriculum Learning (RoCL)

Neural network training can easily overfit noisy labels resulting in poor generalization performance. Existing methods address this problem by (1) filtering out the noisy data and only using the clean data for training or (2) relabeling the noisy data by the model during training or by another model trained only on a clean dataset. However, the former does not leverage the features' information of wrongly-labeled data, while the latter may produce wrong pseudo-labels for some data and

introduce extra noises. In Chapter 13, we propose a smooth transition and interplay between these two strategies as a curriculum that selects training samples dynamically. In particular, we start with learning from clean data and then gradually move to learn noisy-labeled data with pseudo labels produced by a time-ensemble of the model and data augmentations. Instead of using the instantaneous loss computed at the current step, our data selection is based on the dynamics of both the loss and output consistency for each sample across historical steps and different data augmentations, resulting in more precise detection of both clean labels and correct pseudo labels. On multiple benchmarks of noisy labels, we show that our curriculum learning strategy can significantly improve the test accuracy without any auxiliary model or extra clean data.

16.2.1 Related Work of Noisy-Label Learning

In the context of robust learning with noisy labels, label correction methods aim to identify the wrong labels and possibly correct them to get more consistent labels for training. Previous work often apply an extra noise model (directed graphical model [237], conditional random fields [215], neural network [124, 218], knowledge graph [131]) to correct the noisy labels, which often require extra clean data and as well as training/inference of the noise model. Another line of research focuses on loss correction, which modifies the loss or prediction probabilities during training to correct the misinformation from the noisy labels. [169] uses two noise transition (backward and forward) matrices to correct the prediction probabilities. Label Smoothing Regularization [205, 170] alleviates the overfitting to noisy labels by using soft labels instead of one-hot labels. [178] augments the loss with a notion of perceptual consistency. [103] trains a mentor network to reweigh samples duri‘ng the training of a student network. [72] designs a curriculum by ranking the complexity of data using its distribution density in a feature space. [179] proposes a meta-learning algorithm that learns to assign weights to samples based on their gradients in training compared to those of validation data, which requires extra clean data. Co-teaching [76] feeds in the network with the

most confident samples of another network to reduce confirmation bias. [2] generalizes the logistic loss and the exponents in the softmax by applying a temperature to each of them and makes the training more robust to noise. [87] trains a network on noisy labels in the weakly supervised setting and uses it as a regularization term to improve the training on clean data.

Some approaches focus on designing loss functions that have robust behaviors and provable tolerance to label noise. [68] theoretically proves that the Mean Absolute Error(MAE) is a robust loss. The Generalized Cross Entropy [246] uses a negative Box-Cox transformation to obtain a loss function that generalizes MAE and Cross Entropy loss. [228] proposes a Symmetric Cross Entropy that combines Cross Entropy loss and Reverse Cross Entropy loss. [144] proposes a loss normalization method and shows that any loss can be made robust to noisy labels.

RoCL shares similar ideas with some CL methods in that RoCL starts with learning easy and clean samples and gradually moves to hard and noisy ones. RoCL is more related to the loss correction approach in noisy-label learning literature as RoCL generates a curriculum dynamically assigning weight (probability) to each sample. RoCL differs from existing methods in: (1) it only selects a subset of informative and reliable labels for training in each epoch; (2) it is a smooth transition not only from clean data to noisy data but also from supervised learning to self-supervision; (3) it runs multiple episodes of the curriculum to avoid getting in a local minimum dominated by a small set of clean/noisy data or a specific type of loss; (4) it does not assume the availability of an extra set of clean data; (5) it does not require extra computation or any modification to the model.

16.2.2 Experiments

We evaluate RoCL with other approaches for noisy-label learning on three widely used benchmarks, i.e., CIFAR10/100 with two types of synthetic noises (i.e., symmetric and asymmetric), and mini-WebVision [130] (the first 50 classes) containing unknown noises from web labels. Symmetric noise flips each label randomly to an incorrect class with probability ρ (i.e., noise rate), and our

Table 16.5: Accuracy (%) evaluated on WebVision and ILSVRC2012 validation sets for DNNs trained by noisy-label learning methods on mini-WebVision training set (first 50 classes), which contains **real-world web-label noises**.

Val. set	WebVision		ILSVRC2012	
Accuracy	Top-1	Top-5	Top-1	Top-5
F-correct $^{+\star}$	61.12	82.68	57.36	82.36
Decoupling **	62.54	84.74	58.26	82.26
Co-teaching *	63.58	85.20	61.48	84.70
MentorNet **	63.00	81.40	57.80	79.92
MentorMix *‡*	76.00	90.20	72.90	91.10
D2L *	62.68	84.00	57.80	81.36
INCV *	65.24	85.34	61.60	84.98
RoCL (ours) $^{\ddagger\star\dagger\wr}$	**80.04**	**92.68**	**75.81**	**92.28**

experiments cover $\rho = \{0.4, 0.6, 0.8\}$. Asymmetric noise flips the labels within a specific set of classes. For CIFAR10, flipping TRUCK→AUTOMOBILE, BIRD→AIRPLANE, DEER→HORSE, CAT→DOG. In CIFAR100, the 100 classes are grouped into 20 super-classes with each has 5 sub-classes, we then flip each class within the same super-class to the next in a circular fashion with probability ρ. Our experiments cover $\rho = \{0.2, 0.3, 0.4\}$.

Practical Modifications

In the experiments, we follow previous work and apply the techniques below. We will present an ablation study of their effectiveness in Table 16.9.

- We apply the **class-balance regularization** used in [206], which prevents the model from predicting the same class for all the samples within a mini-batch B. We add $(1/B) \sum_{i \in B} \ell(f(x_i; \theta), 1/C \cdot \mathbf{1})$ (where C is the number of classes) to the objective for a mini-batch B with regularization weight of 1.
- We apply **label smoothing** whose effectiveness in noisy-label learning has been studied in [143]. We modify each one-hot label y_i to be $\bar{y}_i \leftarrow (1 - \alpha) y_i + \alpha/C$ (e.g., we use $\alpha = 0.5$).

- We apply **Mix-Up** [243] to all the selected data. However, Mix-Up of two (soft) pseudo labels can significantly increase the entropy of the mixed label if both pseudo labels are under-confident. Hence, we apply a curriculum to the beta distribution's parameter α of Mix-Up and gradually reduce it (e.g., from 8.0 to 0.2 in our experiments) within each episode.

Hyperparameters

We apply RoCL to train ResNet34 on CIFAR10/100 and ResNet50 on WebVision, which are the most widely used models in other baseline papers. We apply SGD with momentum of 0.9, weight decay of 10^{-4} and cosine annealing learning rate in each training episode. The initial learning rate is set to 0.1 for CIFAR10/100 and 1.0 for WebVision. In all RoCL experiments, we apply $T_0 = 10$ warm starting epochs followed by $K = 10$ episodes of curriculum learning, whose lengths start from $T_1 = 10$ and increase by 10 for every episode afterwards. We initialize the subset size $b_0 = 0.2n$ and set $\gamma = \gamma_b = 0.1$, which are common choices for discounting/augmenting factors. We use [46] for data augmentations. We did not heavily tune λ_1, λ_T and τ_1, τ_T and followed a principle that the resulting curriculum should have a transition from supervised learning on clean data to self-supervised learning on noisy data with correct pseudo labels.

- For λ, we start from λ_1 close to 1 and end with λ_T close to 0 because our curriculum is a transition from supervised learning ($\lambda = 1$) to self-supervised learning ($\lambda = 0$).
- As explained in Section 3.3, our curriculum requires τ_1 for $p_t(i)$ changing from negative to positive values and an inverse sequence for τ_2 in $q_t(i)$ to gain the above transition and encourage learning on more informative samples. Hence, we set the starting value τ_1 to be negative and τ_T to be positive for the sequence $\tau_{1:T}$.
- We set their exact values based on observations in Figure 8.20: clean data detection is easier but the detection of correct pseudo-labels is harder. So we can be more confident on the former than the latter and set the starting τ_1 larger than τ_T in magnitude. For the same reason, we set

the starting value λ_1 to be closer to 1 than the ending value λ_T to 0. In experiments, we tried $\tau_1 = \{-4, -3\}$ and $\tau_T = \{1, 2\}$ and finally chose $\tau_1 = -4$ and $\tau_T = 1$ since this choice performs consistently well on all experiments, though it might not be the best choice for all; we set $\lambda_1 = 0.9$ and did not try other values; we tried $\lambda_T = \{0.1, 0.2, 0.3\}$ and on some experiments the first two choices lead to slightly worse performance, so we chose $\lambda_T = 0.3$.

Main Results

We compare RoCL to F-correct [169], Decoupling [147], Co-teaching [76], D2L [145], INCV [36], MD-DYR-SH [4], MentorNet [103], MentorMix [99], O2U-net [89], RoG+D2L [123], PENCIL [239], GCE [246], SCE [228], NFL/NCE variants [144], and Bootstrap [178].

To better compare and categorize different baseline methods, we use the following symbols to denote the techniques used: $+$ for additional clean training data; $*$ for training additional auxiliary models; ‡ for using mixup; $\star$ for using data augmentations; † for class-balance regularization; $\wr$ for label-smoothing. We report the results and comparisons to baselines in three tables: real-world noise in Table. 16.5, symmetric noise in Table. 16.7 and asymmetric noise in Table. 16.8. RoCL achieves the best performance in every setting, and for most of the cases, improves upon the existing methods by large margins. The closest rival to RoCL is MentorMix, which utilizes MentorNet and Mix-Up to assign weights to each sample. We note that MentorMix requires training of an extra mentor network to generate the sample weights, while RoCL is more flexible and only makes changes to the training process

Table 16.6: Test accuracy (%) of **RoCL** applied with different loss functions on CIFAR10 corrupted by $\{60\%, 80\%\}$ **symmetric(uniform)** noises (CE-cross entropy).

Noise Rate	60%	80%
CE	90.22 ± 0.24	77.47 ± 0.67
GCE	89.30 ± 0.68	79.84 ± 1.12
SCE	$\mathbf{92.06 \pm 0.23}$	74.25 ± 0.86
NFL+MAE	88.73 ± 0.47	$\mathbf{85.76 \pm 0.26}$
NFL+RCE	87.68 ± 0.35	80.09 ± 0.41
NCE+MAE	90.37 ± 0.43	82.16 ± 0.93
NCE+RCE	88.03 ± 0.39	80.33 ± 0.80

without modifying the model. Table. 16.6 reports RoCL's performance when applied with different

Table 16.7: Test accuracy (%) of noisy-label learning methods on CIFAR10/100 corrupted by **symmetric(uniform)** label noises of different levels. All the baselines' results are from the original papers or the following-up works. There are two formats of these reported results: "mean±variance" of 5 trials and single-trial accuracy.

Dataset	CIFAR10			CIFAR100		
Noise Rate	40%	60%	80%	40%	60%	80%
MD-DYR-SH **‡†	92.3	86.1	74.1	70.1	59.5	39.5
MentorNet **	91.2	74.2	60.0	68.5	61.2	35.5
MentorMix *‡*	94.2	91.3	81.0	71.3	64.6	41.2
O2U-net *	90.3	-	43.4	69.2	-	39.4
RoG+D2L **	87.0	78.0	-	64.9	40.6	-
PENCIL *	-	-	-	69.12 ± 0.62	57.79 ± 3.86	fail
GCE *	87.62 ± 0.26	82.70 ± 0.23	67.92 ± 0.60	62.64 ± 0.33	54.04 ± 0.56	29.60 ± 0.51
SCE *	85.34 ± 0.07	80.07 ± 0.02	53.81 ± 0.27	53.69 ± 0.07	41.47 ± 0.04	15.00 ± 0.04
NFL+MAE *	83.81 ± 0.06	76.36 ± 0.31	45.23 ± 0.52	58.18 ± 0.08	46.10 ± 0.50	24.78 ± 0.82
NFL+RCE	86.05 ± 0.12	79.78 ± 0.13	55.06 ± 1.08	58.20 ± 0.31	46.30 ± 0.45	25.16 ± 0.55
NCE+MAE	84.19 ± 0.43	77.61 ± 0.05	49.62 ± 0.72	59.22 ± 0.36	48.06 ± 0.34	25.50 ± 0.76
NCE+RCE *	86.02 ± 0.09	79.78 ± 0.50	52.71 ± 1.90	59.48 ± 0.56	47.12 ± 0.62	25.80 ± 1.12
RoCL (ours) $^{\ddagger\star\dagger\wr}$	$\mathbf{94.55 \pm 0.12}$	$\mathbf{92.06 \pm 0.23}$	$\mathbf{85.76 \pm 0.26}$	$\mathbf{74.64 \pm 0.43}$	$\mathbf{66.79 \pm 0.58}$	$\mathbf{53.89 \pm 0.62}$

loss functions on CIFAR10 under high noise rates, i.e., 60% and 80%. We observe significant improvements over their performance without using RoCL in Table. 16.7. It indicates that RoCL is compatible with any loss function and can further enhance their performance.

Table 16.8: Test accuracy (%) of noisy-label learning methods on CIFAR10/100 corrupted by **asymmetric(class-dependent)** noises of 3 levels. All the baselines' results are from the original papers or the following-up works.

Dataset	CIFAR10			CIFAR100		
Noise Rate	20%	30%	40%	20%	30%	40%
PENCIL *	92.43	91.84	91.01	74.70 ± 0.56	72.52 ± 0.38	63.61 ± 0.23
Bootstrap *	86.57 ± 0.08	84.86 ± 0.05	79.76 ± 0.07	63.44 ± 0.35	63.18 ± 0.35	62.08 ± 0.22
F-correct $^{+\star}$	89.09 ± 0.47	86.79 ± 0.36	83.55 ± 0.58	42.46 ± 2.16	38.13 ± 2.97	34.44 ± 1.93
GCE *	86.07 ± 0.31	80.78 ± 0.21	74.98 ± 0.32	59.99 ± 0.83	53.99 ± 0.29	41.49 ± 0.79
SCE *	83.92 ± 0.07	79.70 ± 0.27	78.20 ± 0.03	58.22 ± 0.47	49.85 ± 0.91	42.19 ± 0.19
NFL+MAE *	86.81 ± 0.32	83.91 ± 0.34	77.16 ± 0.10	63.10 ± 0.22	56.19 ± 0.61	43.51 ± 0.42
NFL+RCE	88.73 ± 0.29	85.74 ± 0.22	79.27 ± 0.43	63.12 ± 0.41	54.72 ± 0.38	42.97 ± 1.03
NCE+MAE	86.44 ± 0.23	83.98 ± 0.52	78.23 ± 0.42	62.38 ± 0.60	58.02 ± 0.48	47.22 ± 0.30
NCE+RCE *	88.56 ± 0.17	85.58 ± 0.44	79.59 ± 0.40	62.68 ± 0.79	57.82 ± 0.41	46.79 ± 0.96
RoCL (ours) $^{\ddagger\star\dagger\wr}$	$\mathbf{95.38 \pm 0.21}$	$\mathbf{94.19 \pm 0.28}$	$\mathbf{92.31 \pm 0.35}$	$\mathbf{80.03 \pm 0.34}$	$\mathbf{77.59 \pm 0.45}$	$\mathbf{73.28 \pm 0.83}$

Table 16.9: **Ablation study:** Test accuracy (%) of RoCL variants with one part removed/changed when applied to CIFAR10/100 corrupted by **symmetric(uniform)** label noise.

Dataset	CIFAR10		CIFAR100	
Noise Rate	60%	80%	60%	80%
RoCL: no MixUp	92.98	**88.18**	**69.72**	**58.72**
RoCL: no LabelSmooth	91.94	85.05	62.92	42.95
RoCL: no ClassBalance	**93.08**	74.91	62.66	43.94
RoCL: no RandAugment	86.59	72.35	64.84	44.06
RoCL: no RandSampling	92.31	85.99	64.09	57.00
RoCL: no EMA metrics	92.84	87.79	65.99	53.10
RoCL: $p_t(i) = 1/n$	92.42	86.05	62.69	44.35
RoCL: $q_t(i) = 1/n$	92.59	86.93	64.71	50.79
RoCL: $p_t(i) = q_t(i) = 1/n$	92.07	85.77	64.18	47.88
RoCL$_{Base}$: no curriculum	87.83	66.93	61.84	41.92
RoCL: original version	92.82	88.00	66.79	54.22
MentorMix: +RandAugment	85.45	20.68	52.70	8.02
MentorMix: $_{\text{+RandAugment-MixUp}}$	84.31	38.21	58.31	8.18
MentorMix: original version	91.30	81.00	64.60	41.20

Ablation Study

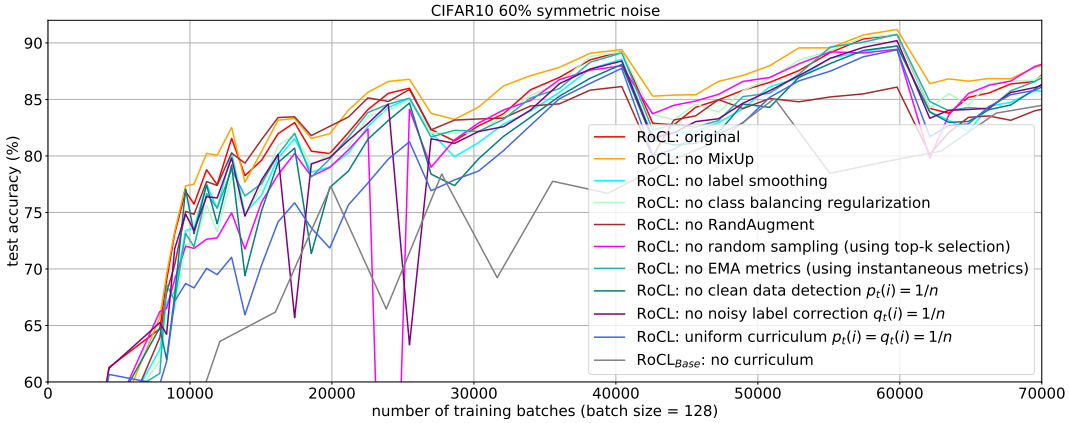

Figure 16.4: **Ablation study**: RoCL vs. its variants during the training of ResNet34 on **CIFAR10** containing **60% symmetric noises** on labels.

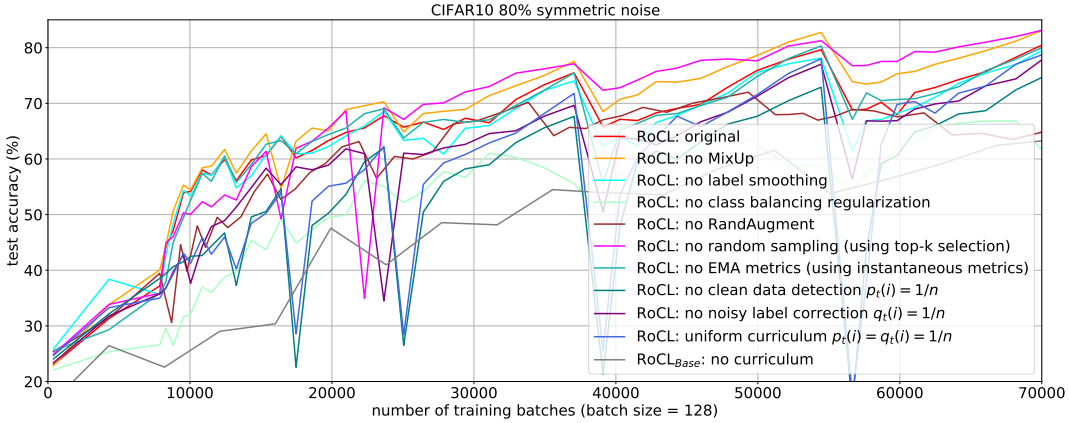

Figure 16.5: **Ablation study**: RoCL vs. its variants during the training of ResNet34 on **CIFAR10** containing **80% symmetric noises** on labels.

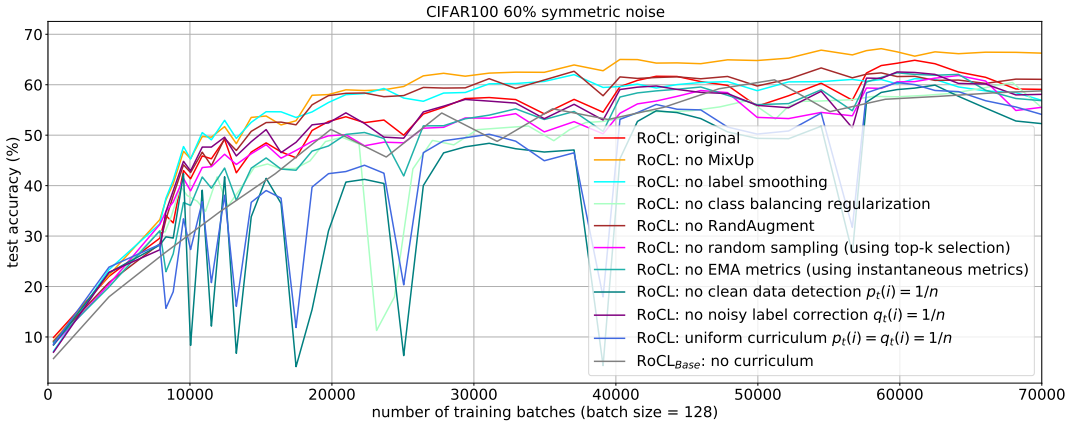

Figure 16.6: **Ablation study**: RoCL vs. its variants during the training of ResNet34 on **CIFAR100** containing **60% symmetric noises** on labels.

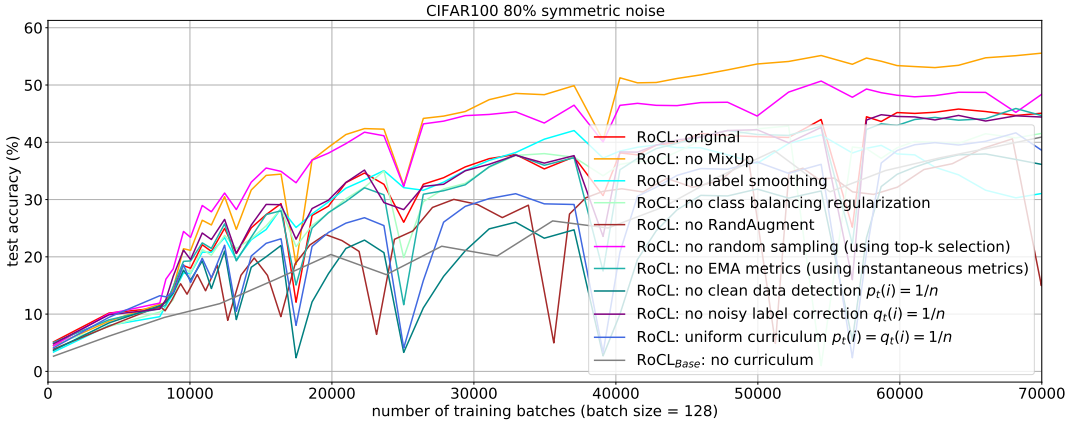

Figure 16.7: **Ablation study**: RoCL vs. its variants during the training of ResNet34 on **CIFAR100** containing **80% symmetric noises** on labels.

To analyze the effect of each component in RoCL, we conduct a thorough ablation study of 10 variants of RoCL, each removing/changing one component of the original RoCL. In Table 16.9, we report their test accuracies on CIFAR10/100 with noise rates of $\{60\%, 80\%\}$. In Figure 16.4-16.7, we report how their test accuracies change during the training to study their learning efficiency and convergence. Among them, "no ClassBalance" removes the class-balance regularization; "no RandAugment" replaces the strong data augmentation RandAugment [46] with random crop and random horizontal flip; "no RandSampling" replaces the weighted sampling in Line 11 of Algorithm 8 by selecting the top-b_k samples with the largest $\mathscr{P}_t(i)$; "no EMA metrics" replaces

EMA loss and EMA consistency loss with their instantaneous counterparts; "$p_t(i) = 1/n$" samples the clean data using uniform probabilities; "$q_t(i) = 1/n$" samples the correct pseudo-labels using uniform probabilities; "$p_t(i) = q_t(i) = 1/n$" uses uniform probabilities for both. Note for the final three variants, we still have the curriculum of λ. We keep the same hyperparameter settings as the original RoCL.

In the following, we provide a detailed analysis of several observations from the ablation study experiments.

- Most variants (except RoCL$_{Base}$, no RandAugment, and no ClassBalance) have similar performance as the original RoCL and perform better than or competitive with the SoTA results achieved by MentorMix. The differences compared to original RoCL become smaller under the lower noise rate setting (60%). RoCL$_{Base}$ uses all data for training in each step without applying any curriculum, showing that our proposed curriculum is the most critical component of RoCL in achieving the appealing improvements. Note RoCL$_{Base}$ already outperforms most methods in Table 16.7, which verifies the effectiveness of multi-episode training that alternates between supervised learning with the given labels and self-supervision with the pseudo labels.
- Removing RandAugment degrades the performance, especially when the noise rate is very high (e.g., 80%) because strong data augmentations are required by the self-supervision and the EMA consistency loss in RoCL, while trivial data augmentations can result in error accumulation or over-confidence in pseudo labels and inaccurate EMA consistency loss. The self-supervision aims to encourage the model output consistency over different augmentations of the same sample. Without augmentations with sufficient variations, self-supervision reduces to reinforcing the same outputs on similar samples and thus carries little information and can even magnify/accumulate errors (if any) in the original outputs. Also, the EMA consistency loss cannot generate meaningful consistency measures if computed on the same data or its trivial augmentations. Note a strong data augmentation is not always beneficial in all noisy label learning methods since it can increase

the uncertainty in the presence of wrong labels, making the detection of clean data and noise correction more challenging. For example, we tried applying RandAugment to MentorMix (using the official implementations of both) but observed inferior performance compared to the results using its original data augmentations.

- Class balance regularization is useful for the very high noise rate setting (80%), in which a wrong label may dominate the learning on a mini-batch by a large chance. However, when the noise rate is not that high (e.g., 60% on CIFAR10), removing it results in better performance.
- Although Mix-Up has been proved effective in previous methods, and for this reason, we followed MentorMix by starting with a relatively strong Mix-Up ($alpha = 8.0$) and then gradually reducing it to $\alpha = 0.2$. In the ablation study, we find that completely removing Mix-Up significantly improves performance. Mix-Up is helpful when applied to mix a clean label with a noisy label since the latter can be mediated with the former and thus softened. However, this is rarely the case for RoCL since RoCL either mainly learns from clean data or wrongly-labeled data with correct pseudo labels, and the transition between the two phases is short. When applied to two correct labels/pseudo labels, Mix-Up weakens each label's confidence, and we may lose information from the inter-class probabilities in the soft pseudo labels.
- Replacing weighted sampling with top-k selection ("no RandSampling") or replacing EMA metrics with instantaneous metrics ("no EMA metrics") causes less degeneration on the final test accuracies. However, they are important to the early-stage exploration and accurate estimation of EMA metrics on less-visited samples. In Figure 16.4-16.7, these two variants usually suffer from low accuracy and convergence speed during early stages. The only exception is "no RandSampling" in Figure 16.7, which performs better than the original RoCL. A possible reason is that the randomness brought by high uniform label noises already bring sufficient randomness for exploration.
- Replacing $p_t(i)$, $q_t(i)$ or both with uniform probabilities over all samples reduces the final test

accuracies in all cases, e.g., the degradation is significant on CIFAR100 with 80% noise. In Figure 16.4-16.7, we can see that by setting $q_t(i) = 1/n$ results in less degradation than the other two. This is due to the more accurate pseudo labels generated for more data (even the ones with larger EMA consistency loss) as training proceeds. Moreover, since we are conservative in setting λ_T and τ_T, the performance is not very sensitive to wrong pseudo labels.

Comparisons between RoCL and RoCL$_{Base}$

In order to show the importance and contribution of the curriculum proposed in RoCL, in the following, we compare RoCL with RoCL$_{Base}$ (no curriculum) over different benchmark settings used in experiments above. The results show that the curriculum brings significant improvement in addition to the two data selection criteria in RoCL$_{Base}$.

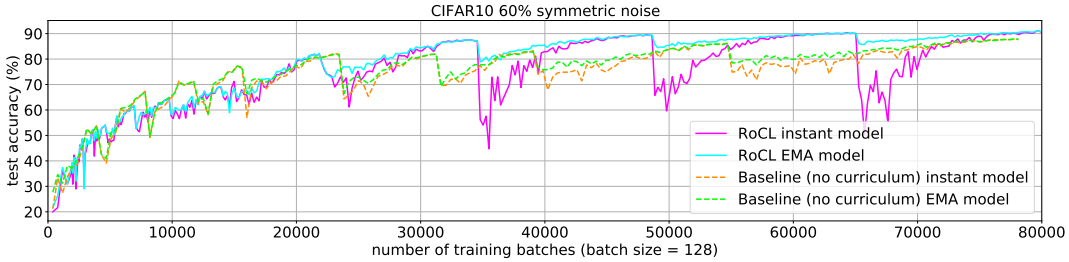

Figure 16.8: RoCL (Algorithm 8) vs. RoCL$_{Base}$ without any curriculum (Algorithm 1) during the training of ResNet34 on **CIFAR10** containing **60% symmetric noises** on labels.

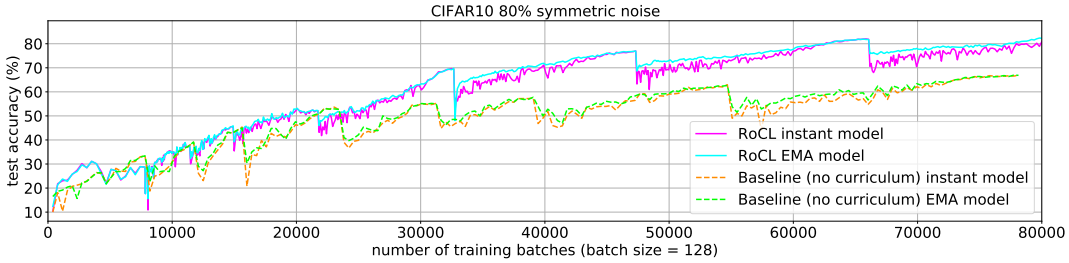

Figure 16.9: RoCL (Algorithm 8) vs. RoCL$_{Base}$ without any curriculum (Algorithm 1) during the training of ResNet34 on **CIFAR10** containing **80% symmetric noises** on labels.

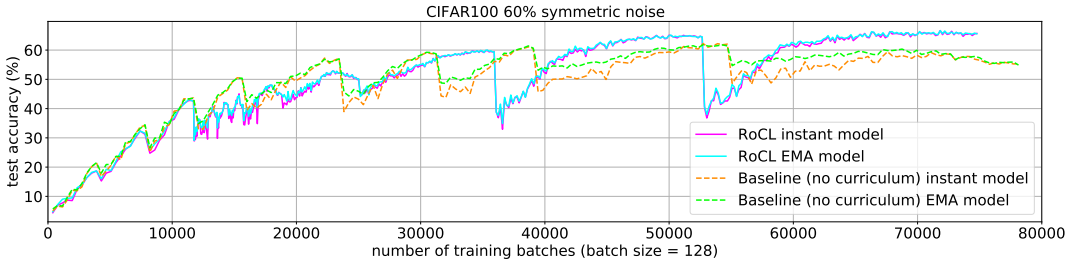

Figure 16.10: RoCL (Algorithm 8) vs. RoCL$_{Base}$ without any curriculum (Algorithm 1) during the training of ResNet34 on **CIFAR100** containing **60% symmetric noises** on labels.

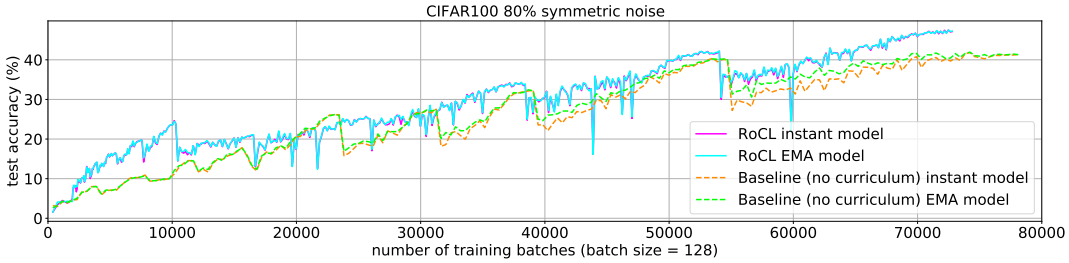

Figure 16.11: RoCL (Algorithm 8) vs. RoCL$_{Base}$ without any curriculum (Algorithm 1) during the training of ResNet34 on **CIFAR100** containing **80% symmetric noises** on labels.

Chapter 17

DIVERSE ENSEMBLE LEARNING

We apply DivE^2 algorithm proposed in Chapter 14 to four benchmark datasets, and show that it improves over randomization-based ensemble training methods on a variety of approaches to aggregate ensemble models into a single prediction. Moreover, with model selection based ensemble aggregation (defined below), DivE^2 can quickly reach reasonably good ensemble performance after only a few learning stages even though each individual model has poor performance on the entire training set. Furthermore, DivE^2 exhibits competitive efficiency and good of model expertise interpretability, both of which can be important in DNN training.

17.1 Experimental Setting

We apply three different ensemble training methods to train ensembles of neural networks with different structures on four datasets, namely: (1) MobileNetV2 [183] on CIFAR10 [114]; (2) ResNet18 [81] on CIFAR100 [114]; (3) CNNs with two convolutional layers* on Fashion-MNIST ("Fashion" in all tables) [236]; (4) and lastly CNNs with six convolutional layers on [40]†. The three training methods include DivE^2 and two widely used approaches as baselines, which are

- Bagging(BAG)[28]: sample a new training set of the same size as the original one (with replacement) for each model, and train it for several epochs on the sampled training set.
- RandINIT(RND): randomly initialize model weights of each model, and train it for several epochs on the whole training set.

Details can be found in Table 17.3 of. We everywhere fix the number of models at $m = 10$,

*A variant of LeNet5 with 64 kernels for each convolutional layer.

†The network structure is from https://github.com/aaron-xichen/pytorch-playground.

and use ℓ_2 parameter regularization on w with weight 1×10^{-4}. In DivE^2's training phase, we start from $k = 6, p = {}^{n}/_{2m}$ and linearly change to $k = 1, p = {}^{3n}/_{m}$ in $T = 200$ episodes. We employ the "facility location" submodular function [42, 136] for both the intra and inter-model diversity, i.e., $F(A) = \sum_{v' \in V} \max_{v \in V(A)} \omega_{v,v'}$ where $\omega_{v,v'}$ represents the similarity between sample v and v'. We utilize a Gaussian kernel for similarity using neural net features $z(v)$ for each v, i.e., $\omega_{v,v'} = \exp\left(-{}^{\|z(v)-z(v')\|^2}/_{2\sigma^2}\right)$, where σ is the mean value of all the ${}^{n(n-1)}/_{2}$ pairwise distances. For every dataset, we train a neural networks on a small random subset of training data (e.g., hundreds of samples) for one epoch, and use the inputs to the last fully connected layer as features z. These features are also used in the Top-k DCS-KNN approach (below) to compute the pairwise ℓ_2 distances to find the K nearest neighbors.

17.2 Aggregation Methods using an Ensemble of Models

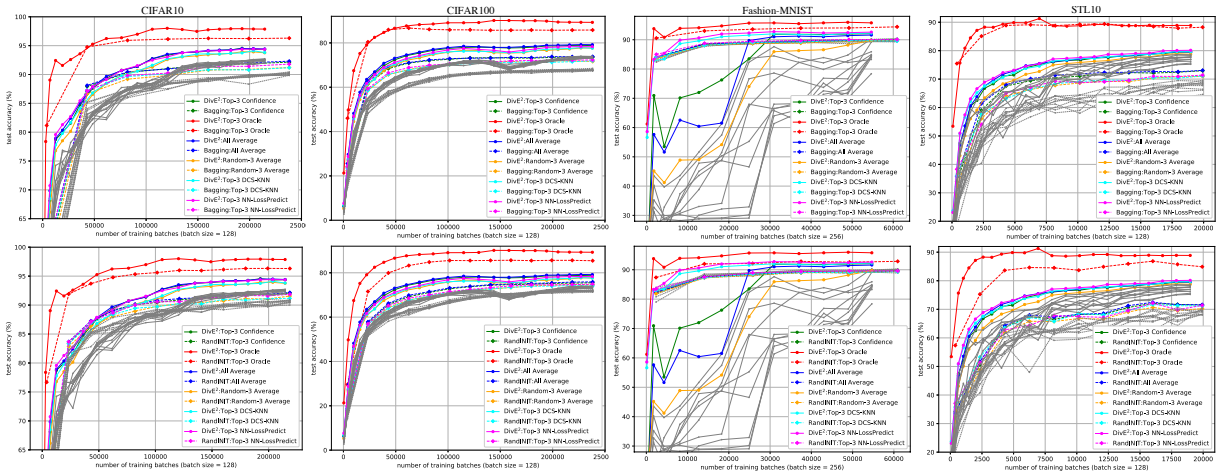

Figure 17.1: Compare DivE^2 with Bagging(upper row) and RandINIT(lower row) in terms of test accuracy (%) vs. number of training batches on CIFAR10, CIFAR100, Fashion-MNIST and STL10, with $m = 10$ and $k = 3$.

For ensemble model aggregation, when applying a trained ensemble of models to new samples, we must determine (1) which models to use, and (2) how to aggregate their outputs. Here we mainly discuss the first point about different model selection methods, because the aggregation we employ

is either an evenly or a weighted average of the selected model outputs. Static model selection methods [257, 33, 167] choose a subset of models from the ensemble and apply it to all samples. By contrast, dynamic classifier selection (DCS) [34, 151, 262, 44] selects different subsets of models to be aggregated for each sample. KNN based DCS [235, 110] is a widely used method that usually achieves better performance than other DCS and static methods. When training, DivE^2 assigns different subsets of samples to different models, so for aggregation, we may benefit more from using sample-specific model selection methods. Therefore, we focus on DCS-type methods, in particular, the following:

- Top-k Oracle: average the outputs (e.g., logits before applying softmax) of the top-k models with the smallest loss on the given sample. It requires knowing the true label, and thus is a cheating method that cannot be applied in practice. However, it shows a useful upper bound on the other methods that select k models for aggregation.
- All Average: evenly average the outputs of all m models.
- Random-k Average: randomly select k models and average their outputs.
- Top-k Confidence: select the top-k models with the highest confidence (i.e., highest probability of the predicted class) on the given sample, and average their outputs.
- Top-k DCS-KNN: apply an KNN based DCS method, i.e., find the K nearest neighbors of the given sample from the training data, select the top-k models assigned to the K nearest neighbors by Top-k Oracle, and average their outputs.
- Top-k NN-LossPredict: train an L2-regression neural nets with m outputs to predict the per-sample losses on the m models by using a training set composed of all training samples and their losses on the trained models. For aggregation, select the top-k models with the smallest predicted losses on the given sample, and average their outputs.

17.3 Experimental Results

We compare the three training methods used with the aforementioned aggregation methods with different k[‡]. We summarize the highest test-set accuracy when $k = 3$ in Table 17.2, and show how the test accuracy improves as training proceeds (i.e., as the total training batches on all models increases) in Figure 17.1. In Figure 17.1, solid curves denote DivE^2, while dashed curves denote the three baseline training methods. Different colors refer to different aggregation methods, and gray curves represent single model performance (gray solid curves denote models trained by DivE^2, while gray dashed curves denote models trained by other baselines). Similar results for $k = 5$ and $k = 7$ can be found in Figure 17.3-17.6. In addition, we also tested DivE^2 without the "model selecting sample" constraint and any diversity, which equals to [125, 73, 126] in multi-class case. It achieves a test accuracy of 90.11% (vs. 94.36% of DivE^2) on CIFAR10 and 71.01% (vs. 78.89% of DivE^2) on CIFAR100 when using Top-3 NN-LP for aggregation.

Table 17.1: Total time (secs.) of DivE^2 and time only on SUBMODULARMAX.

Dataset	CIFAR10	CIFAR100	Fashion	STL10
Total time	26790.75s	34658.27s	2922.89s	4065.81s
SUBMODULARMAX	1857.36s	2697.36s	81.64s	378.84s

Top-k Oracle (cheating) is always the best, and provides an upper bound. In addition, DivE^2 usually has higher upper bound than others, and thus has more potential for future improvement. Solid curves (DivE^2) are usually higher than dashed curves (other baselines) in later stages, no matter which aggregation method is used. Although diversity introduces more difficult samples and lead to slower convergence in early stages, it helps accelerate convergence in later stages. Although the test accuracy on single models achieved by DivE^2 is usually lower than those obtained by other baselines, the test accuracy on the ensemble is better. This indicates that different models indeed develop different local expertise. Hence, each model performs well good only in a local

[‡]The k used in aggregation fixed, and is different from the k in training (which decreases from 6 to 1).

Table 17.2: The highest test accuracy (%) achieved by different combinations of ensemble training and aggregation methods on four datasets, with $k = 3$. DivE^2 usually requires less training time than others to achieve the highest accuracy. The best non-cheating test accuracy (i.e., not Top-k Oracle) is highlighted below.

Train:Aggregation	CIFAR10	CIFAR100	Fashion	STL10
BAG:Top-k Oracle (Cheat)	97.85	88.02	95.60	89.13
BAG:All Average	93.69	73.12	91.24	74.96
BAG:Random-k Avg.	93.05	72.86	91.00	74.03
BAG:Top-k Confidence	93.51	74.59	90.81	75.76
BAG:Top-k DCS-KNN	92.86	73.06	91.39	74.07
BAG:Top-k NN-L.P.	93.45	73.62	92.38	75.16
RND:Top-k Oracle (Cheat)	97.80	87.01	95.71	89.54
RND:All Average	93.28	75.71	91.13	77.13
RND:Random-k Avg.	93.11	75.56	90.77	76.75
RND:Top-k Confidence	93.51	76.54	91.07	77.93
RND:Top-k DCS-KNN	93.18	75.72	92.01	77.23
RND:Top-k NN-L.P.	93.69	76.69	92.48	77.28
DivE^2:Top-k Oracle (Cheat)	98.01	90.12	96.40	90.18
DivE^2:All Average	94.20	**79.12**	86.16	78.95
DivE^2:Random-k Avg.	93.26	77.69	82.75	78.59
DivE^2:Top-k Confidence	94.05	78.76	92.10	79.38
DivE^2:Top-k DCS-KNN	93.81	77.61	92.10	79.23
DivE^2:Top-k NN-L.P.	**94.36**	78.89	**92.76**	**80.49**

region but poorly elsewhere. However, their expertise is complementary, so the overall performance of the ensemble outperforms other baselines. We visualize the expertise of each model across different classes in Figure 17.2 for Fashion-MNIST as an example. Among all aggregation methods, Top-k NN-LossPredict and Top-k DCS-KNN show comparable or better performance than other aggregation methods, but require much less aggregation costs when k is small. As shown in Figure 17.3-17.6, when changing k from minority ($k = 3$) to majority ($k = 7$), the test accuracy of these two aggregation methods usually improves by a large margin. According to Table 17.1, DivE^2 only requires a few extra computational time for data assignment. The model training dominates the computations but is highly parallelizable since the updates on different models are independent.

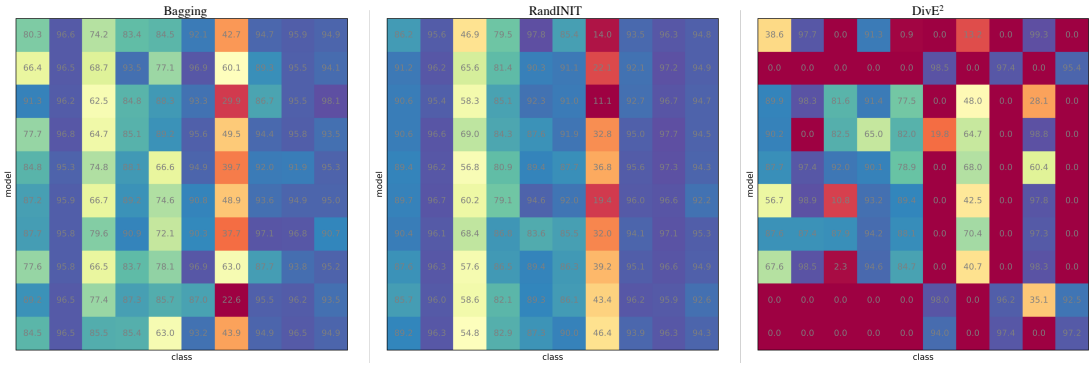

Figure 17.2: Test accuracy (%) per class on each single model from the ensemble trained by Bagging(left, after 18750 total training batches), RandINIT(middle, after 18750 total training batches) and DivE^2 (right, after 18249 total training batches) on Fashion-MNIST. This figure reflects the expertise of each model on different classes. Comparing to Bagging and RandINIT, the models learned by DivE^2 show diverse and complementary expertise.

Table 17.3: Details regarding the datasets.

Dataset	CIFAR10	CIFAR100	Fashion	STL10
#Training	50000	50000	60000	5000
#Test	10000	10000	10000	8000
#Feature	$3 \times 32 \times 32$	$3 \times 32 \times 32$	28×28	$3 \times 96 \times 96$
#Class	10	100	10	10

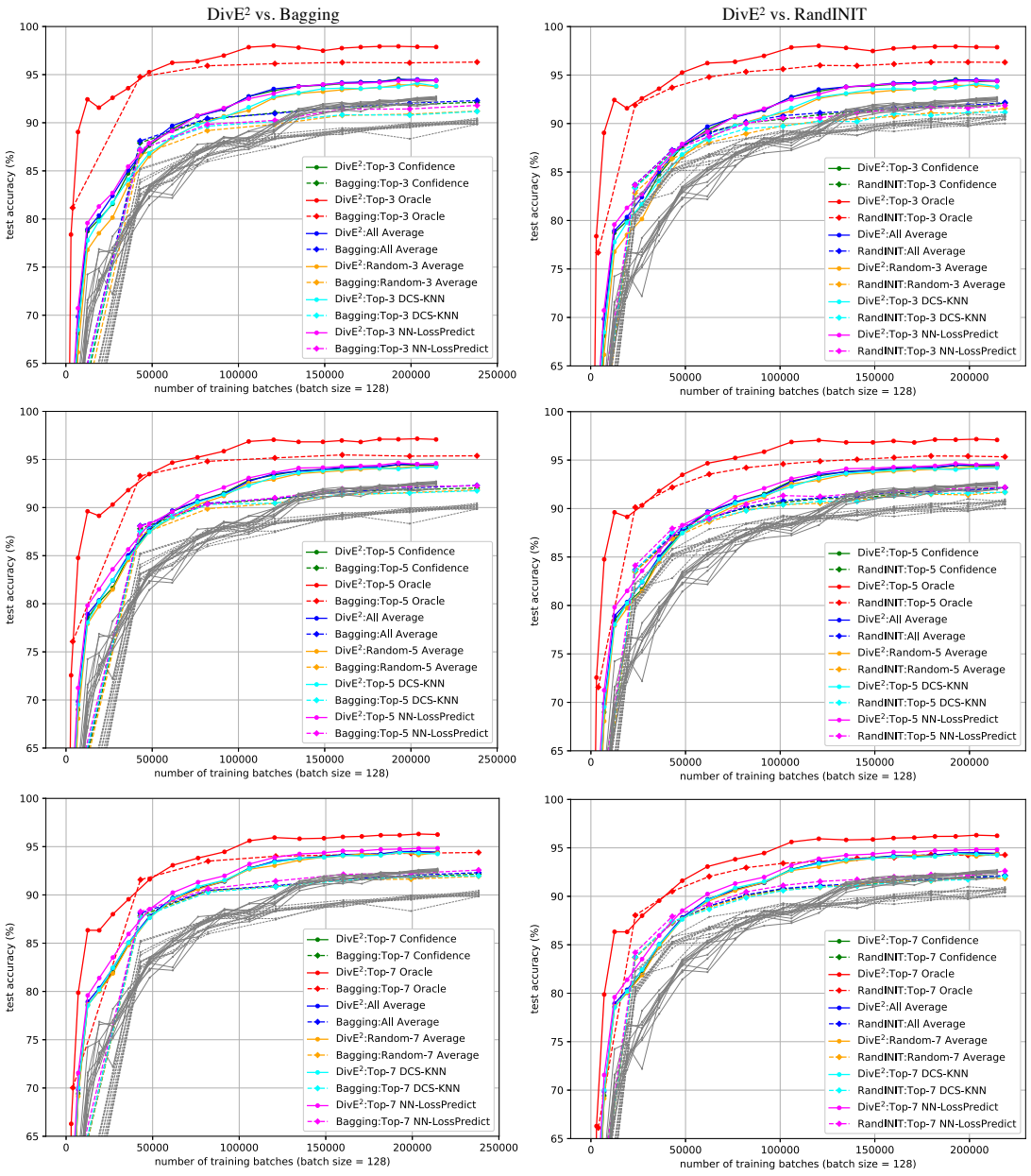

Figure 17.3: Compare DivE^2 with Bagging(left column) and RandINIT(right column) in terms of test accuracy (%) vs. number of training batches on CIFAR10, with $m = 10$ MobileNetV2 models trained, and using different k values ($k = 3, 5, 7$ from top to bottom) for aggregation.

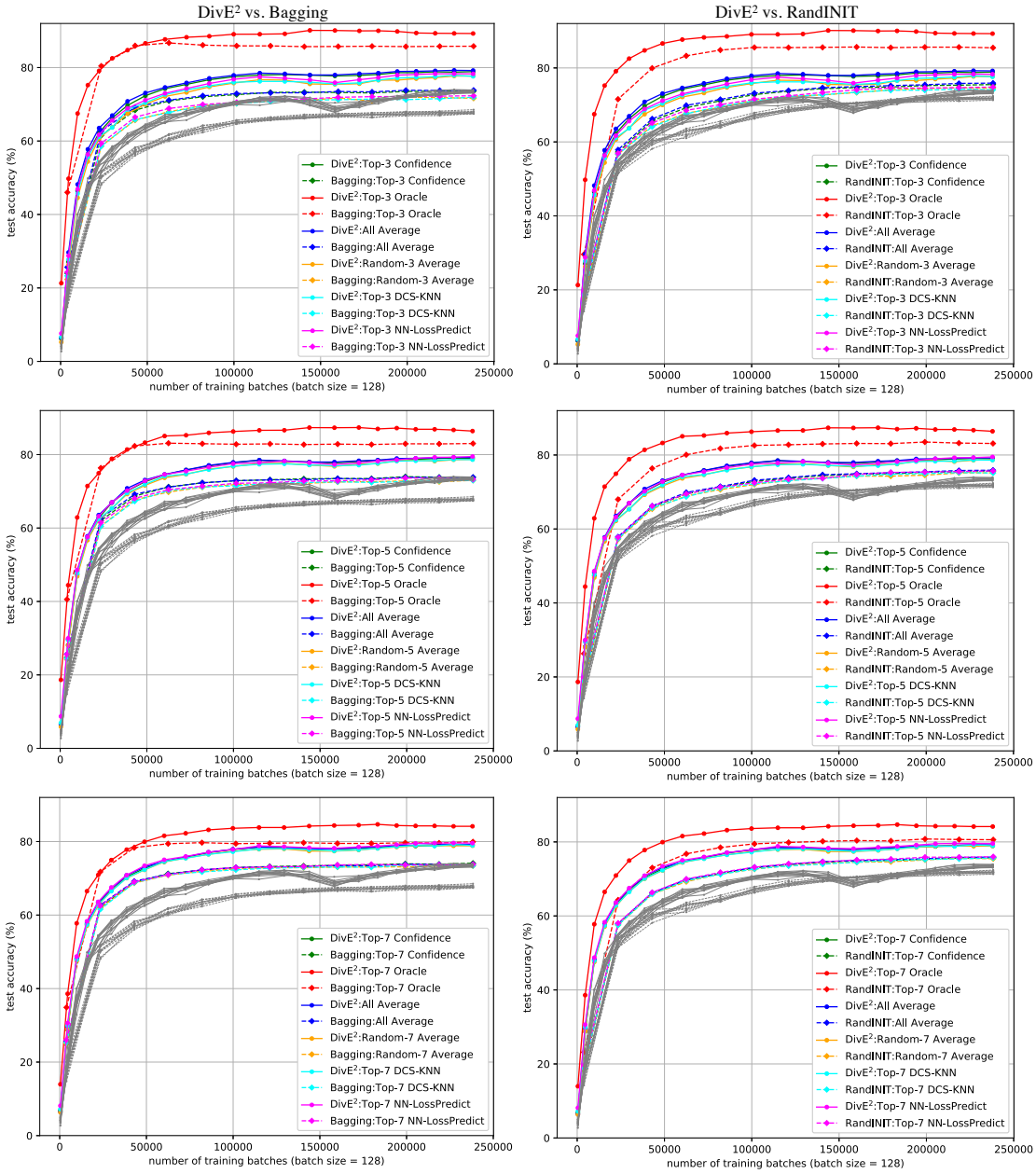

Figure 17.4: Compare DivE^2 with Bagging(left column) and RandINIT(right column) in terms of test accuracy (%) vs. number of training batches on CIFAR100, with $m = 10$ ResNet18 models trained, and using different k values ($k = 3, 5, 7$ from top to bottom) for aggregation.

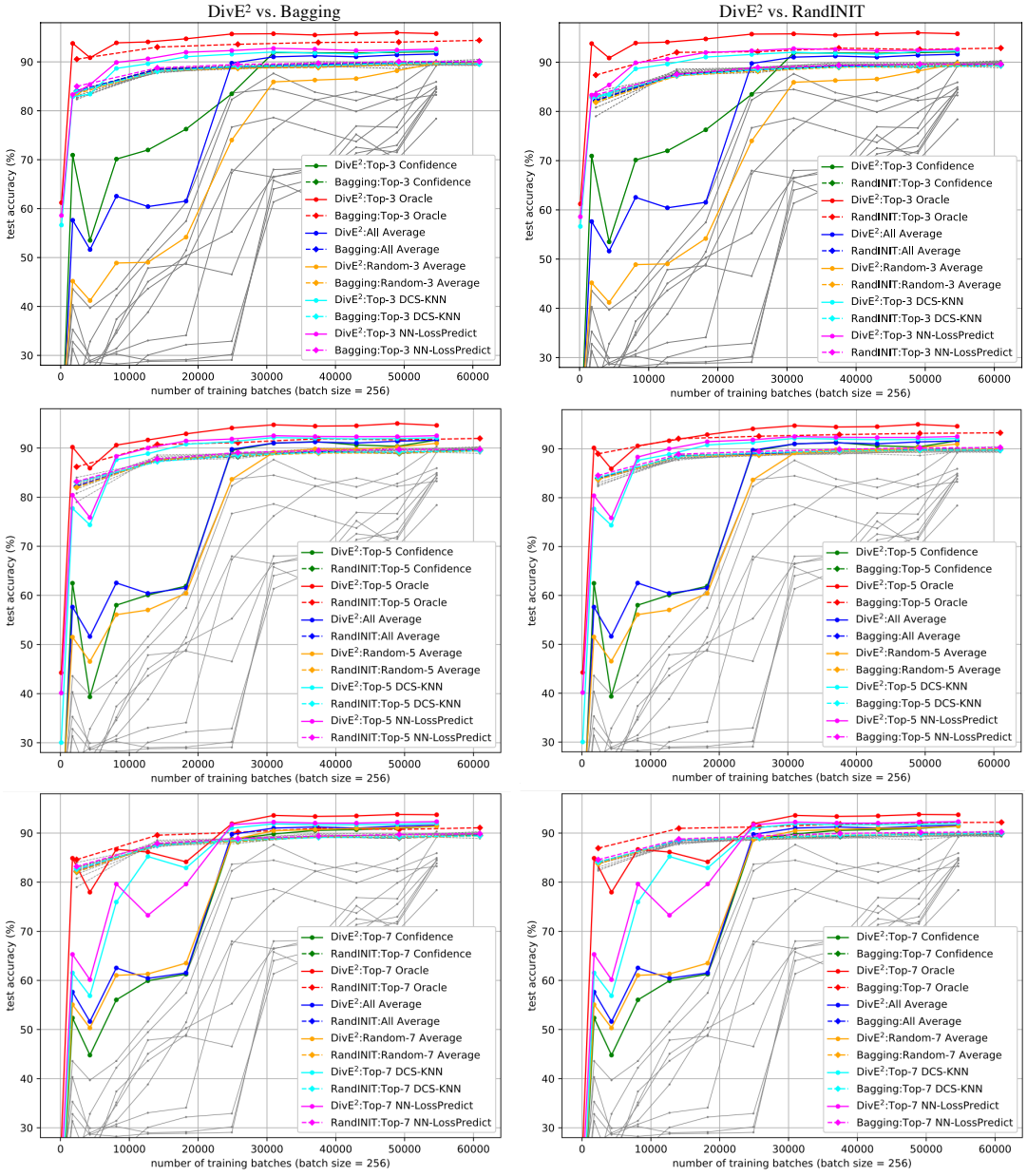

Figure 17.5: Compare DivE^2 with Bagging(left column) and RandINIT(right column) in terms of test accuracy (%) vs. number of training batches on Fashion-MNIST, with $m = 10$ modified LeNet5 models trained, and using different k values ($k = 3, 5, 7$ from top to bottom) for aggregation.

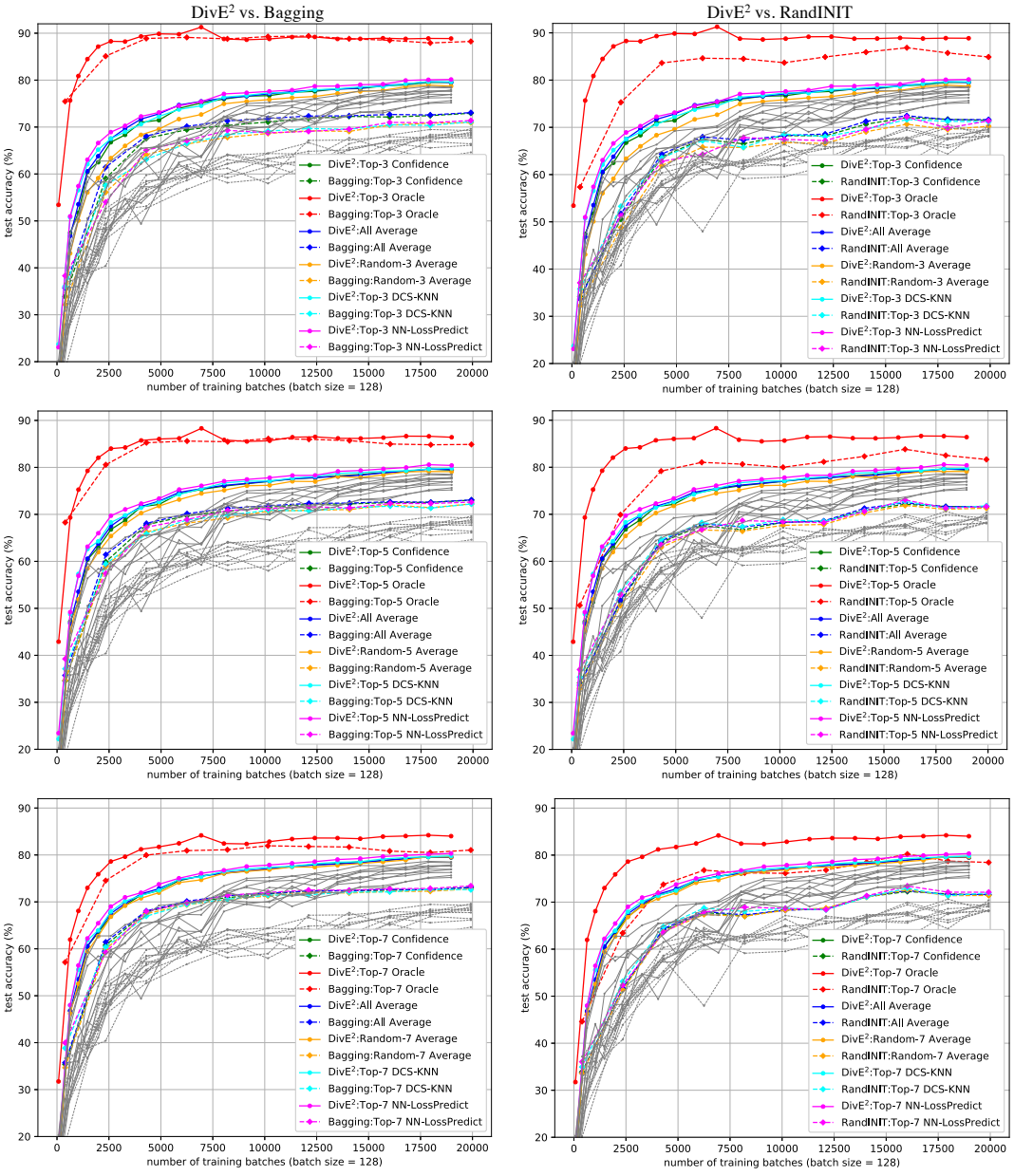

Figure 17.6: Compare DivE^2 with Bagging(left column) and RandINIT(right column) in terms of test accuracy (%) vs. number of training batches on STL10, with $m = 10$ CNN models trained, and using different k values ($k = 3, 5, 7$ from top to bottom) for aggregation.

Chapter 18

CONCLUSIONS AND FUTURE DIRECTIONS

18.1 Summary of Contributions

This thesis studies how to automatically design a curriculum, e.g., a sequence of training data or tasks, to improve the learning efficiency and generalization performance of ML models. Curriculum learning aims at bridging a critical gap between machine learning and human learning. While human learning can significantly benefit from training contents and strategies optimized for different learning stages based on feedback from the learning process and/or teachers' experiences, the currently mainstream paradigms for machine learning pre-determine those training data/tasks/models/hyperparameters, fix them over the course of training, and repeat them for multiple stages. This is highly sub-optimal and inefficient according to human learning strategies, which are good at adjusting training plans and teachers' guidance in order to achieve powerful generalization even with deficient data or time.

The main contributions of this thesis can be summarized as the following:

Novel problem formulations for curriculum learning. Instead of developing curriculum learning methods solely based on heuristics, we studied several novel optimization formulations in Part I, from which we can derive practical curriculum learning algorithms with theoretical motivations or properties in later chapters, e.g., the convergence analysis of MCL in Section 9.4, the analysis of catastrophic forgetting in Section 8.3.1, the analysis to learning dynamics and connections to NTK in Section 8.2, the theoretical properties of DiVE2 in Section 14.2, etc. In Chapter 2, we discussed the continuous-discrete hybrid optimization formulation where the discrete variable refers to the

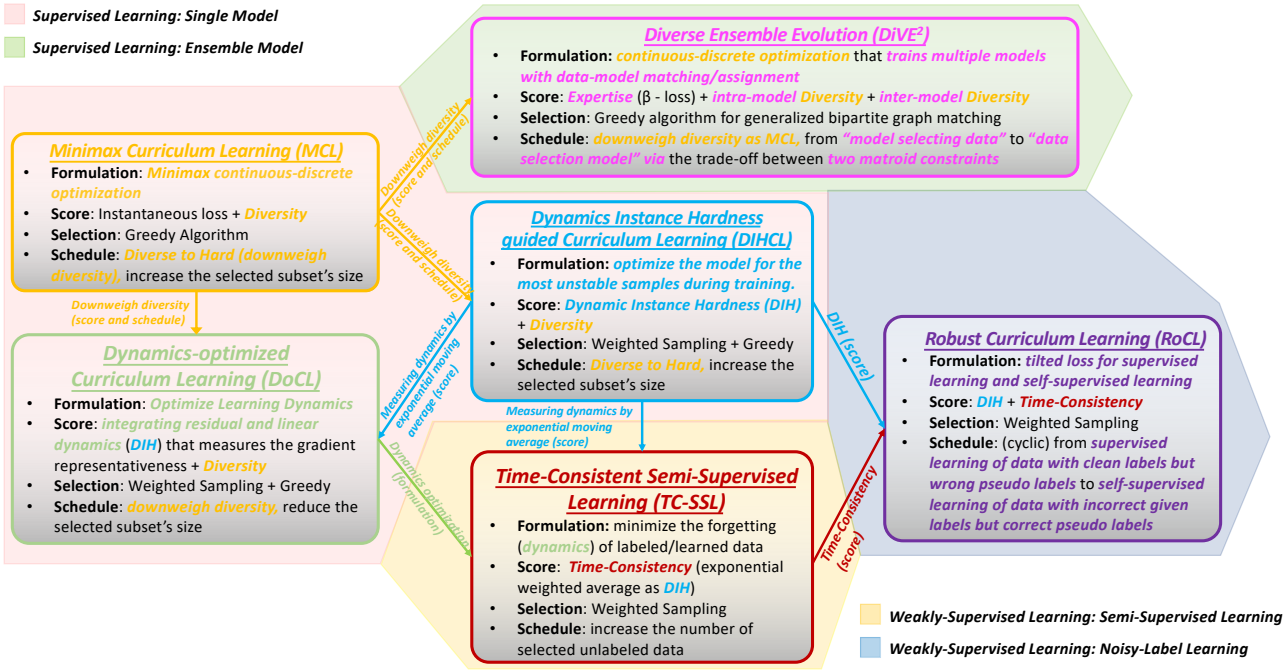

Figure 18.1: Summary of the six curriculum learning methods proposed in this thesis: (1) For each method, we briefly summarize its four components, i.e., its formulation, score, selection criterion, and scheduling strategy; (2) We highlight the main novelty of each method using the color for the name of the method; (3) We illustrate the connections between different methods and the ideas/techniques shared between them; (4) The background color of each method represent its targeted machine learning problem.

subset of training data selected by the curriculum in each stage. We proposed a novel minimax formulation and compared it with the existing min-min formulation. Chapter 3 introduces a new formulation of curriculum learning based on the tilted loss [129], which can reweigh each sample in training according to its loss values and a controllable temperature. As an example, we show that the formulation can be utilized to build a noisy-label learning curriculum targeting at two different objectives. In Chapter 4, we formulate the optimization of a curriculum as optimizing the training dynamics achieved on a selected subset of training data in continuous time. The formulation results in a family of score functions that integrate several heuristics adopted in existing effective curriculum learning methods. In Chapter 5, we studied the curriculum learning for an ensemble of models and formulate it as a continuous-discrete hybrid optimization, where the discrete part solves a data-model matching/assignment problem for each learning stage. In Chapter 6, we discussed other possible formulations worth exploring in the future.

Score functions capturing diversity and training dynamics. Another primary contribution of this thesis is the score functions proposed in Part II. We discover that two new families of score functions are effective in allocating the most informative training data and thus can be critical to curriculum learning. The first class of scores aim at capturing the diversity and redundancy over a subset of selected elements so it can further improve the sample efficiency of curriculum learning by avoiding selecting the ones carrying similar information. We show that a rich class of well studied set functions, i.e., submodular functions, can accurately measure the diversity scores and their theoretical properties usually result in provable curriculum learning algorithms. The second class of scores are built upon the training dynamics and historical statistics on each sample or task exposed in the training process. There are several significant advantages of these training dynamics based scores when compared with the widely studied instantaneous feedback: (1) they provide higher-quality and more precise indicators of important training materials because they

are more consistent, stable, and suffer from smaller variance over time, while the instantaneous feedback of neural network training is usually unstable and noisy; (2) their low-cost estimations are computed from the by-products of training without incurring any extra computational cost, while the instantaneous scores have to be re-evaluated for each candidate sample or task in every step even they will not be selected for training; (3) they provide new patterns describing the learning process over time and the flatness of the loss landscape, which are critical to build more effective and human-like curricula; (4) they are derived from or results in the optimization of training dynamics, which provides a principal objective for curriculum learning that naturally relates to deep neural network theories, e.g., Neural Tangent Kernel (NTK). We developed score functions based on training dynamics for labeled data, unlabeled data, and data with noisy labels. Moreover, we design and present thorough empirical studies of the scores, whose results demonstrated their advantages over previous scores and paved the way towards more efficient and effective curriculum learning approaches.

Novel curriculum learning algorithms. In Part III, we develop practical curriculum learning algorithms based on the new formulations and equipped with the new scores proposed in Part I and Part II. In the algorithmic designs, besides selecting higher-scored data with priority, we additionally apply different exploration strategies, which is essential to the quality of scores estimated from the training dynamics because only the scores of those selected samples can be updated in each step. Moreover, we propose smooth scheduling of selection criteria changing across different learning stages in order to achieve curricula specifically designed for each problem. These algorithms are developed for a broad class of representative machine learning problems: (1) three algorithms for supervised learning, i.e., minimax curriculum learning (MCL), dynamic instance hardness guided curriculum learning (DIHCL), and dynamics-optimized curriculum learning (DoCL); (2) time-consistent semi-supervised learning for semi-supervised learning; (3) robust curriculum learning

for noisy-label learning; and (4) diverse ensemble evolution (DivE^2) for ensemble learning. For each problem, we discussed and investigated a rich class of practical techniques and tricks that have been demonstrated to be effective in previous work and evaluate their effectiveness in our curriculum learning setting. Moreover, we also provide theoretical analyses or insights to most proposed algorithms.

Applications and experiments of the proposed curriculum learning algorithms and comparisons to baselines on benchmark datasets. For each proposed curriculum learning algorithm, we conduct extensive experiments, ablation studies, and comparisons to the corresponding baselines (or recently reported results) in different settings on a variety of challenging and widely-used benchmark datasets. The experimental results demonstrate the advantages of our proposed algorithms over competitive baselines and their improvements on both the time/sample efficiency and the generalization performance on hold-out test sets. In addition, for each proposed algorithm, we provide a thorough empirical analysis to interpret the reasons behind its improvements brought by the proposed curriculum. In particular, in Chapter 16, we observe a phenomenal potential of curriculum learning on improving ML in the wild with imperfect data and weak supervision, e.g., self/semi-supervised learning and noisy-label learning. Existing ML methods can easily fail or perform unstable if selecting low-quality data for training, while previous methods achieving the SoTA performance are notoriously compute-intensive with slow convergence. Moreover, we extend curriculum learning for training a single-model to training an ensemble of multiple expert models. In the experiments, our proposed DivE^2 algorithm can quickly train an ensemble of diverse but complementary models, each only focusing on a different data region. This is achieved through a data-model matching curriculum. The techniques and observations of DivE^2 in ensemble learning paves the way towards collaborative/cooperative curriculum learning on a structured network of devices/nodes/agents, as an imitation of school education, workspace training, or cultural evolution

of humans.

18.1.1 User Guidelines of Proposed Methods

As summarized above, this thesis focuses on addressing several critical challenges for curriculum learning, e.g., how to build accurate and efficient score functions from historical statistics of training dynamics, how to formulate the joint optimization of an ML model and a curriculum, how to design a scheduling strategy that can produce an efficient learning plan across multiple stages, how to determine the selection criterion for different learning tasks or settings? By studying these problems, we attempt to bridge the gaps among practical algorithms, heuristics from human cognition, principal optimization formulations, and theoretical insights. In particular, we develop six curriculum learning approaches in this thesis, three (MCL, DIHCL, DoCL) for supervised learning of a single model, one (DiVE^2) for ensemble learning of multiple diverse and complementary models, one (TCSSL) for semi-supervised learning, and one (RoCL) for noisy-label learning. For supervised learning, as shown in the experiments of Section 15.3, DoCL outperforms MCL and DIHCL over a variety of standard benchmark datasets. The other three approaches address three different machine learning problems, respectively. So one can choose which method to apply according to the problem setting in practice. Although the six approaches are different on their targeted problems, formulations, scores, selection criteria, or scheduling strategies, they share some core ideas or methodologies. Moreover, some of them are built upon the key discoveries of the others. In Figure 18.1, we provide a summary of the key components of each approach, the connections and shared ideas between different approaches, and their targeted machine learning problems.

18.1.2 Connections between Proposed Methods

We start from the most widely studied machine learning problem, i.e., supervised learning of a single model. Inspired by the focus of representative hard examples in human learning, we propose

a minimax formulation of curriculum learning in Section 2.2 and the associated MCL algorithm in Chapter 9, which starts from training with a few diverse samples and gradually introduce more samples as well as increasing the preference of hard samples. In order to overcome the drawbacks of instantaneous hardness scores, in Chapter 10 and Eq. 10.2, we replace the loss-based hardness with the dynamic instance hardness (DIH) that measures the hardness of each sample by the sharpness of its training dynamics. As a result, the DIHCL algorithm in Chapter 10 empirically achieves better efficiency and test accuracy on a variety of benchmarks. However, in DIHCL and most previous work, the mathematical connections between the data selection criterion the targeted empirical risk minimization are not straightforward. Hence, in Chapter 4, we study a curriculum that searches for the subset of training data (of limited size) that can result in the greatest improvement of loss on the whole data distribution. This dynamics-optimization reduces to selecting samples according to the DoCL score introduced in Section 8.2, which, surprisingly, is consistent in spirit with the DIH and diversity scores, i.e., it tends to select hard samples whose model outputs change fast over time and whose gradients are representative of other samples drawn from the distribution. Moreover, we build connections of the DoCL score with neural tangent kernel (NTK) [95] for deep learning theories in Section 8.2.4.

In order to extend curriculum learning from single-model training to ensemble learning of multiple expert models, we replace data selection in each learning stage with data-model matching, i.e., how to assign training data to different models, and how to assign models to each sample. We propose DivE^2 in Chapter 5 and Chapter 14, which can efficiently learn a diverse yet complementary ensemble by a smooth transition from "model selecting data" to "data selecting model". DivE^2 achieves this by gradually changing the parameters in two matroid constraints and considering the diversity of training data for a single model and for every two models. By emphasizing the diversity scores (intra-model and inter-model diversity) in the data-model assignment during earlier stages and downweighing it later on, similar to MCL, DivE^2 enforces earlier expansion of expertise per

model and the diversity among models.

We then study curriculum learning for two weakly-supervised learning problems, in which the selection of training data is more significant to the learning efficiency and performance. Firstly, we extend the idea of measuring the prediction consistency over time in DIHCL to unlabeled data and study the time-consistency of their model predictions. In particular, following the methodology of dynamics optimization in DoCL, we study how to minimize the forgetting dynamics of the learned (and labeled) samples in semi-supervised learning by selecting unlabeled data according to their time-consistency scores. This leads to TC-SSL for semi-supervised learning in Chapter 12. Secondly, by combining the time-consistency score in TC-SSL for selecting unlabeled data and DIH in DIHCL for selecting labeled data, we develop a curriculum learning method addressing the noisy-label learning problem. RoCL introduced in Chapter 13 is able to train a model using the most informative samples with high-quality labels or pseudo labels through a curriculum from supervised learning to self-supervised learning.

18.2 Open Challenges and Future Directions

Although we addressed some fundamental challenges of curriculum learning in this thesis, as summarized above. There still exist a variety of open problems that have not been fully explored and following-up challenges of the problems studied in this thesis. In the following, we will discuss several important future research topics and potential directions.

Bridging the training dynamics and geometry of DNNs with curriculum generation. Although most existing curriculum learning methods are inspired by human learning, reverse engineering of human intelligence on machines is usually neither optimal nor feasible. To certain extents, A curriculum's success mainly depends on how well it is optimized for the training dynamics and geometric structure of the ML model. In our previous work, we connected the data/task selection criterion with the training dynamics via empirical and theoretical analysis. Moreover, we related

the curriculum's objective to recent deep learning theories. We have also studied the geometry of ReLU DNNs as polyhedra associated with linear models and how the training process changes the geometry, which inspires better dropout [225] and interpretation methods [224]. However, there are still a variety of open problems and fundamental questions. For example, how to measure the forgetting effects on learned data/tasks via training dynamics and control it by curriculum design so we can develop better continual learning with life-long knowledge accumulation? How, why, and when should we encourage diversity, curiosity, or other human-learning heuristics in the optimization of training dynamics on data? Which physical systems may we follow when optimizing the integrated energy of the training dynamics and what convergence rates can we achieve? How to measure the order complexity of a curriculum and what is its influence to the generalization error?

Curriculum learning beyond data selection: task-level, domain (environment)-level, and model-level curriculum learning. As discussed in Chapter 1, curriculum learning is not only limited to the scheduling of hyperparameters or sequential data selection, which are unfortunately its most common forms in existing literature. In practice, there are still many critical open challenges for ML that can be significantly benefited from higher-level curricula, for example, (1) how to improve the generalization and transferability of ML models to new domains or tasks? how to foster different fundamental skills or learn universal representations that can be shared across different tasks/domains? (2) how to decompose a difficult task into multiple subtasks that are easier and more efficiently to learn by ML models in a specific order? (3) how to improve ML models or agents' robustness to distribution shift or environment change? How to perform stability-plasticity trade-off and avoid catastrophic forgetting in a life-long learning setting? (4) how to progressively grow or prune an ML model such as neural networks in a data and task-driven manner? how to optimize the morphology of an intelligent agent to adapt to new tasks or environments? These problems have been discussed in transfer learning, multi-task learning, meta-learning, continual

learning, neural network pruning, model compression, neural architecture search, reinforcement learning and robotics, etc., but they are still open challenges without ideal solutions. By manipulating the order of training tasks/environments/domains and automatically determining the model architecture/morphology to explore, curriculum learning on different levels can provide a natural (human-like) and novel methodology to solve these critical problems.

Exploring new formulations of curriculum learning. A primary challenge for the current curriculum learning research is the deficiency of rigorous formulations that have both strong theoretical motivations and practical algorithms. Besides the four types of formulations introduced in Chapter 2-5, in Chapter 6, we discussed several possible formulations for curriculum learning that can be potentially built upon multi-armed bandits (MAB), reinforcement learning (RL), and meta-learning in the future. Although these formulations are based on classical problems that have been studied in machine learning for decades, new challenges arise when crafted or modified specifically for curriculum learning. For example, we need to modify multi-armed bandit settings to capture the non-stationary nature of multi-stage curriculum learning and to scale to a large amount of (possibly combinatorial) arms; when formulated as a reinforcement learning problem, the consistency within each stage and the smoothness between contiguous stages of a curriculum are usually preferred but how to determine the length of each stage is an open challenge; for both the RL and meta-learning formulations, since the curriculum is learned from multiple learning experiences or tasks, it is essential to reduce their costs (or lengths) and improve the curriculum's transferrability to other learning tasks.

Optimization challenges for curriculum learning in continual-discrete optimization and game settings. One fundamental yet challenging problem in developing curriculum learning algorithms is the optimization of an objective containing both continual variables (e.g., model parameters) and discrete variables (e.g., a selected subset of data). In Section 2.2 and Chapter 5, we studied

two examples of this type of optimization for two special cases of curriculum learning, in which submodular optimization plays important roles in formulating the objectives and constraints for the curriculum. However, more general cases and probably a unified theory need to be studied for other curriculum learning settings. For example, in collaborative curriculum learning, the curriculum needs to determine both the partitioning of models and a fair data selection, which can be cast into certain forms of constrained robust submodular optimization [226]. Moreover, when considering a joint curriculum of multiple levels (e.g., data-level and task-level together), more complicated constraints are needed for the discrete optimization part. Furthermore, it is still an open problem to analyze the equilibrium of a game between a non-convex optimization player and a discrete optimization player. It is also unclear whether an algorithm(s) can achieve the optimal solutions or equilibrium provably.

Curriculum self-supervised/representation learning for foundation models. Self-supervised learning is a key component of human-level intelligence. In recent years, large-scale foundation models produced by self-supervised learning becomes popular and turn out to be powerful few-shot or even zero-shot learner for a great number of down-stream tasks. For many practical ML tasks, only a very limited amount of labeled data is available or affordable so representations learned by self-supervised from unlabeled or even out-of-domain data are essentially helpful. In Chapter 12 and Chapter 13, we developed two curriculum learning algorithms, i.e., TC-SSL [251] and RoCL [256], which relates the consistency of model outputs over training epochs with the utility of unlabeled data for self-supervised learning objectives such as pseudo-labeling. A more general challenge is to automatically generate a sequence of self-supervision tasks on selected data in order to improve the representation learning or the downstream task adaptation. This raises several new challenges, e.g., how to parameterize the tasks and evaluate them using feedback from the training history. Moreover, the correlation between self-supervision tasks and the target tasks may also play an important role

in the curriculum generation. While most existing self-supervised learning methods train a model on one or two tasks selected by human experts, the auto-curriculum of tasks can potentially bring more improvements. This idea can be further extended to transfer learning, multi-task learning, and meta-learning for more efficient knowledge transfer and distillation across tasks.

Collaborative curriculum learning among multiple models/agents on a network . In curriculum learning, we usually consider a teacher-student interaction between a curriculum generator and an ML model. In DiVE2 [252], we studied classroom-type learning, in which a teacher assigns different training contents to multiple students. However, human learning is more intricate and effective in more sophisticated scenarios. We can find several types of collaborative group learning in the university system, where some of the most important developments in science, technology, and the humanities have taken place. For example, students benefit from in-class teacher-present learning and teacher-free learning within study groups, where multiple students gather and raise questions of, and deduce answers from each other. We can develop analogous ML techniques that allow collaborative learning, mutual knowledge distillation, adversarial training, and other peer-to-peer learning among multiple ML models with heterogeneous architectures and data. This raises challenges in new mathematical objectives, optimization algorithms, and a curriculum that decides the subset of models and their communication contents in every step. This technique can lead to more efficient multi-agent learning, federated learning, decentralized learning, meta-learning, and edge AI on sensor networks or mobile devices.

Catastrophic forgetting and knowledge transfer/sharing in curriculum learning. Curriculum learning has natural connections to several key challenges of other fields such as continual learning, transfer learning, multi-task learning, etc. Since a curriculum is composed of a sequence of learning stages with an optimized or designed order, a fundamental requirement for the learning order is that the catastrophic forgetting, if any, should not be harmful to the final targeted task(s). Moreover, the

knowledge transferred from an earlier stage of a curriculum is expected to improve or accelerate its later learning stages. Furthermore, the resulted knowledge accumulation over stages needs to be more effective and efficient than other possible orders. Hence, the planning and scheduling in curriculum learning need to take into account the task similarity and transferrability, which are also studied in transfer learning and multi-task learning. Moreover, it is even interesting to study a new class of curriculum learning adopting a hybrid setting of continual learning and multi-task learning, in which a learning stage can either focus on a single task or a selected subset of collaborating tasks.

Generative and adversarial curriculum learning. This thesis mainly studied curriculum learning that selects a subset of existing data samples for each training stage. However, automatic perturbation or generation of data can considerably expand the design space of curricula and potentially improve their quality. Moreover, adversarial perturbation can precisely allocate the vulnerability and weakness of a model during training, which provide critical information for curriculum design. Furthermore, a generative model can be trained or fine-tuned in an end-to-end manner with the student model to automatically produce the most informative data during the course of training. Combining the strength of both adversarial attacks and generative models may lead to a principal approach searching for training data in a continuous latent space of data manifold. Empowered by recent progress on powerful generative models such as generative adversarial network (GAN), variational autoencoder (VAE), flow/diffusion based generative models, etc., generative and adversarial curricula might be an alternative to selective curricula in certain scenarios, e.g., when there are plenty of unlabeled data to train a generative model or a pre-trained generative model presumably exists.

BIBLIOGRAPHY

[1] E. L. Allgower and Kurt Georg. *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics, 2003.

[2] Ehsan Amid, Manfred KK Warmuth, Rohan Anil, and Tomer Koren. Robust bi-tempered logistic loss based on bregman divergences. In *Advances in Neural Information Processing Systems*, pages 15013–15022, 2019.

[3] Anonymous Anonymous. Supplementary material for minimax curriculum learning. In *Submitted to ICLR*, 2018.

[4] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. *arXiv preprint arXiv:1904.11238*, 2019.

[5] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems 32*, pages 8141–8150. 2019.

[6] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 233–242, 2017.

[7] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio,

Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *ICML*, volume 70, pages 233–242, 2017.

[8] Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. There are many consistent explanations of unlabeled data: Why you should average. In *International Conference on Learning Representations*, 2019.

[9] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2003.

[10] Wenruo Bai, Jeffrey Bilmes, and William S. Noble. Bipartite matching generalizations for peptide identification in tandem mass spectrometry. In *7th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM BCB)*, ACM SIGBio, Seattle, WA, October 2016. ACM, ACM SIGBio.

[11] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. In *Proceedings of the 23rd International Conference on Machine Learning*, page 65–72, 2006.

[12] Sumit Basu and Janara Christensen. Teaching classification boundaries to humans. In *AAAI*, pages 109–115, 2013.

[13] Dhruv Batra, Payman Yadollahpour, Abner Guzman-Rivera, and Gregory Shakhnarovich. Diverse m-best solutions in markov random fields. In *ECCV*, pages 1–16, 2012.

[14] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear programming - theory and algorithms (2. ed.)*. Wiley, 1993.

[15] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embed-

ding and clustering. In *Advances in Neural Information Processing Systems 14 (NeurIPS)*, pages 585–591. 2002.

[16] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[17] Yoshua Bengio. *Evolving Culture Versus Local Minima*, pages 109–138. Springer Berlin Heidelberg, 2014.

[18] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. *Label Propagation and Quadratic Criterion*, pages 193–216. Semi-supervised learning edition, 2006.

[19] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48, 2009.

[20] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L. Alexander, David W. Jacobs, and Peter N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, 2014.

[21] David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *International Conference on Learning Representations (ICLR)*, 2020.

[22] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 5050–5060. 2019.

[23] Ellen Bialystok, Fergus I. M. Craik, and Gigi Luk. Bilingualism: consequences for mind and brain. *Trends in Cognitive Sciences*, 16(4):240–250, 2012.

[24] Jeff A. Bilmes. Submodularity in machine learning and artificial intelligence. *CoRR*, abs/2202.00132, 2022.

[25] Jeffrey A. Bilmes and Wenruo Bai. Deep submodular functions. *CoRR*, abs/1701.08939, 2017.

[26] Robert A Bjork and Elizabeth L Bjork. A new theory of disuse and an old theory of stimulus fluctuation. *From learning processes to cognitive processes: Essays in honor of William K. Estes*, 2:35–67, 1992.

[27] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014.

[28] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[29] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[30] Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, 2006.

[31] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 535–541, 2006.

[32] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[33] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, 2004.

[34] Paulo R. Cavalin, Robert Sabourin, and Ching Y. Suen. Dynamic selection approaches for multiple classifier systems. *Neural Computing and Applications*, 22(3):673–688, 2013.

[35] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems 30*, pages 1002–1012. 2017.

[36] Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. *arXiv preprint arXiv:1905.05040*, 2019.

[37] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607, 2020.

[38] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, page 539–546, 2005.

[39] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. 2018.

[40] Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pages 215–223, 2011.

[41] Michele Conforti and Gerard Cornuejols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discrete Applied Mathematics*, 7(3):251–274, 1984.

[42] G. Cornuéjols, M. Fisher, and G.L. Nemhauser. On the uncapacitated location problem. *Annals of Discrete Mathematics*, 1:163–177, 1977.

[43] Andrew Cotter, Mahdi Milani Fard, Seungil You, Maya Gupta, and Jeff Bilmes. Constrained interacting submodular groupings. In *International Conference on Machine Learning (ICML)*, Stockholm, Sweden, July 2018.

[44] Rafael M.O. Cruz, Robert Sabourin, and George D.C. Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41:195–216, 2018.

[45] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

[46] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[47] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[48] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, pages 746–751, 2005.

[49] Ido Dagan and Sean P. Engelson. Committee-based sampling for training probabilistic classifiers. In *ICML*, pages 150–157, 1995.

[50] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *ICML*, pages 208–215, 2008.

[51] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[52] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[53] Diana Dolmans and Henk Schmidt. The advantages of problem-based curricula. *Postgraduate medical journal*, 72, 1996.

[54] Pedro Domingos. Every model learned by gradient descent is approximately a kernel machine, 2020.

[55] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[56] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[57] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–451, 2004.

[58] Gamaleldin F. Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alex Kurakin, Ian J. Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both human and computer vision. *arXiv*, 2018.

[59] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *International Conference on Learning Representations*, 2018.

[60] Farzan Farnia and David Tse. A minimax approach to supervised learning. In *NeurIPS*, pages 4240–4248, 2016.

[61] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 1998.

[62] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[63] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions-II. *Mathematical Programming Studies*, 8, 1978.

[64] Madalina Fiterau and Artur Dubrawski. Projection retrieval for classification. In *Advances in Neural Information Processing Systems 25*, pages 3023–3031. 2012.

[65] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[66] Satoru Fujishige. *Submodular functions and optimization*. Annals of discrete mathematics. Elsevier, 2005.

[67] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, pages 174–188, 2011.

[68] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. *arXiv preprint arXiv:1712.09482*, 2017.

[69] Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-optimal map inference for determinantal point processes. In *NeurIPS*, pages 2735–2743, 2012.

[70] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems 17 (NeurIPS)*, pages 529–536. 2005.

[71] Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1311–1320, 2017.

[72] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R Scott, and Dinglong Huang. Curriculumnet: Weakly supervised learning from large-scale web images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018.

[73] Abner Guzmán-rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In *Advances in Neural Information Processing Systems 25*, pages 1799–1807. 2012.

[74] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2535–2544, 2019.

[75] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *ICML*, 2019.

[76] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*, pages 8527–8537, 2018.

[77] Shizhong Han, Zibo Meng, AHMED-SHEHAB KHAN, and Yan Tong. Incremental boosting convolutional neural network for facial action unit recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 109–117. 2016.

[78] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[79] Cédric Hartland, Sylvain Gelly, Nicolas Baskiotis, Olivier Teytaud, and Michèle Sebag. Multi-armed Bandit, Dynamic Environments and Meta-Bandits. 2006.

[80] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *ArXiv*, abs/1911.05722, 2019.

[81] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[82] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 558–567, 2019.

[83] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[84] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[85] C.E. Hmelo-Silver. Problem-based learning: What and how do students learn? *Educational Psychology Review*, 16:235–266, 2004.

[86] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282, 1995.

[87] Mengying Hu, Hu Han, Shiguang Shan, and Xilin Chen. Weakly supervised image classification through noise regularization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11517–11525, 2019.

[88] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get M for free. In *International Conference on Learning Representations (ICLR)*, 2017.

[89] Jinchi Huang, Lie Qu, Rongfei Jia, and Binqiang Zhao. O2u-net: A simple noisy label detection approach for deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3326–3334, 2019.

[90] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Label propagation for deep semi-supervised learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5065–5074, 2019.

[91] R. Iyer and J. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. *In UAI*, 2012.

[92] Rishabh Iyer and Jeff Bilmes. Submodular point processes with applications in machine learning. In *AISTATS*, May 2015.

[93] Rishabh Iyer, Stefanie Jegelka, and Jeff A. Bilmes. Fast semidifferential-based submodular function optimization. In *ICML*, 2013.

[94] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computing*, 3(1):79–87, 1991.

[95] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and

generalization in neural networks. In *Advances in Neural Information Processing Systems 31*, pages 8571–8580. 2018.

[96] S. Jegelka and J. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

[97] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9865–9874, 2019.

[98] Angela H. Jiang, Daniel L. K. Wong, Giulio Zhou, David G. Andersen, Jeffrey Dean, Gregory R. Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C. Lipton, and Padmanabhan Pillai. Accelerating deep learning by focusing on the biggest losers, 2019.

[99] Lu Jiang, Di Huang, Mason Liu, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. ICML, 2020.

[100] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander G. Hauptmann. Self-paced learning with diversity. In *NeurIPS*, pages 2078–2086, 2014.

[101] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. Self-paced curriculum learning. In *AAAI*, pages 2694–2700, 2015.

[102] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2304–2313, 2018.

[103] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning

data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313, 2018.

[104] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computing*, 6(2):181–214, 1994.

[105] K J Joseph, Vamshi Teja R, Krishnakant Singh, and Vineeth N Balasubramanian. Submodular batch selection for training deep neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2677–2683, 2019.

[106] Faisal Khan, Xiaojin (Jerry) Zhu, and Bilge Mutlu. How do humans teach: On curriculum learning and teaching dimension. In *NeurIPS*, pages 1449–1457, 2011.

[107] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[108] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[109] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13):3521–3526, 2017.

[110] Albert H. R. Ko, Robert Sabourin, and Alceu Souza Britto, Jr. From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, 41(5):1718–1731, 2008.

[111] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes

Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, pages 282–293, 2006.

[112] Ágnes Melinda Kovács and Jacques Mehler. Flexible learning of multiple speech structures in bilingual infants. *Science*, 325(5940):611–612, 2009.

[113] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[114] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[115] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 231–238. 1995.

[116] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NeurIPS*, pages 1189–1197, 2010.

[117] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[118] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*, 2017.

[119] Gert R.G. Lanckriet, Laurent El Ghaoui, Chiranjib Bhattacharyya, and Michael I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research (JMLR)*, 3:555–582, 2003.

[120] Ken Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995.

[121] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[122] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 2013.

[123] Kimin Lee, Sukmin Yun, Kibok Lee, Honglak Lee, Bo Li, and Jinwoo Shin. Robust inference via generative classifiers for handling noisy labels. *arXiv preprint arXiv:1901.11300*, 2019.

[124] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5447–5456, 2018.

[125] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David J. Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv*, abs/1511.06314, 2015.

[126] Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. In *Advances in Neural Information Processing Systems 29*, pages 2119–2127. 2016.

[127] Sebastian Leitner. Leitner system. 1970.

[128] Ping Li, Jennifer Legault, and Kaitlyn A. Litcofsky. Neuroplasticity as a function of second language learning: Anatomical changes in the human brain. *Cortex*, 58:301–324, 2014.

[129] Tian Li, Ahmad Beirami, Maziar Sanjabi, and Virginia Smith. Tilted empirical risk minimization. *arXiv preprint arXiv:2007.01162*, 2020.

[130] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017.

[131] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. Learning from noisy labels with distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1910–1918, 2017.

[132] Zhongyuan Li and Sanjeev Arora. An exponential learning rate schedule for deep learning. *ArXiv*, abs/1910.07454, 2019.

[133] Junwei Liang, Lu Jiang, Deyu Meng, and Alexander Hauptmann. Learning to detect concepts from webly-labeled video data. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 1746—-1752, 2016.

[134] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *ACL*, pages 510–520, 2011.

[135] Hui Lin and Jeff Bilmes. Word alignment via submodular maximization over matroids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 170–175. Association for Computational Linguistics, 2011.

[136] Hui Lin, Jeff Bilmes, and Shasha Xie. Graph-based submodular selection for extractive summarization. In *Proc. IEEE Automatic Speech Recognition and Understanding (ASRU)*, 2009.

[137] Ji Liu and Xiaojin Zhu. The teaching dimension of linear learners. *Journal of Machine Learning Research*, 17(162):1–25, 2016.

[138] Weiyang Liu, Bo Dai, Ahmad Humayun, Charlene Tay, Chen Yu, Linda B. Smith, James M. Rehg, and Le Song. Iterative machine teaching. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2149–2158, 2017.

[139] Weiyang Liu, Bo Dai, Xingguo Li, Zhen Liu, James Rehg, and Le Song. Towards black-box iterative machine teaching. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3141–3149, 2018.

[140] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory (TIT)*, 28(2):129–137, 1982.

[141] I. Loshchilov and F. Hutter. Online batch selection for faster training of neural networks. In *International Conference on Learning Representations (ICLR) 2016 Workshop Track*, May 2016.

[142] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[143] Michal Lukasik, Srinadh Bhojanapalli, Aditya Krishna Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? *arXiv preprint arXiv:2003.02819*, 2020.

[144] Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Erfani, and James Bailey. Normalized loss functions for deep learning with noisy labels. *arXiv preprint arXiv:2006.13554*, 2020.

[145] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah M Erfani, Shu-Tao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. *arXiv preprint arXiv:1806.02612*, 2018.

[146] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

[147] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, pages 960–970, 2017.

[148] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[149] Mark McDaniel and Andrew C. Butler. A contextual framework for understanding when difficulties are desirable. *In A. S. Benjamin (Ed.), Successful remembering and successful forgetting: A festschrift in honor of Robert A. Bjork*, pages 175—-198, 2011.

[150] Andrea Mechelli, Jenny T. Crinion, Uta Noppeney, John O'Doherty, John Ashburner, Richard S. Frackowiak, and Cathy J. Price. Neurolinguistics: Structural plasticity in the bilingual brain. *Nature*, 431(7010):757–757, 2004.

[151] Christopher J. Merz. *Dynamical Selection of Learning Algorithms*, pages 281–290. Springer New York, 1996.

[152] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, volume 7 of *Lecture Notes in Control and Information Sciences*, chapter 27, pages 234–243. Springer Berlin Heidelberg, 1978.

[153] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1812–1818, 2015.

[154] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization. *Journal of Machine Learning Research*, 17(238):1–44, 2016.

[155] T. Miyato, S. Maeda, M. Koyama, and S. Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2019.

[156] Mohammad Moghimi, Mohammad Saberian, Jian Yang, Li-Jia Li, Nuno Vasconcelos, and Serge Belongie. Boosted convolutional neural networks. In *British Machine Vision Conference (BMVC)*, 2016.

[157] M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *UAI*, 2005.

[158] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[159] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2004.

[160] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[161] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[162] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, 2018.

[163] Partha Niyogi. Manifold regularization and semi-supervised learning: Some theoretical analyses. *Journal of Machine Learning Research*, 14:1229–1250, 2013.

[164] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[165] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 3235–3246. 2018.

[166] Michael R. Osborne, Brett Presnell, and Berwin A. Turlach. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000.

[167] Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. Focused ensemble selection: A diversity-based method for greedy ensemble selection. In *European Conference on Artificial Intelligence (ECML)*, pages 117–121, 2008.

[168] Kaustubh R Patil, Xiaojin Zhu, Ł ukasz Kopeć, and Bradley C Love. Optimal teaching for limited-capacity human learners. In *NeurIPS*, pages 2465–2473, 2014.

[169] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952, 2017.

[170] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[171] Adarsh Prasad, Stefanie Jegelka, and Dhruv Batra. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *NeurIPS*, pages 2645–2653, 2014.

[172] Ricardo B. C. Prudencio, Jose Hernandez-Orallo, and Adolfo Martinez-Uso. Analysis of instance hardness in machine learning using item response theory. In *2nd International Workshop on Learning over Multiple Contexts (LMCE 2015)*, 2015.

[173] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, page 759–766, 2007.

[174] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[175] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[176] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems 28*, pages 3546–3554. 2015.

[177] R. Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97:285–308, 1990.

[178] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[179] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018.

[180] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations (ICLR)*, 2015.

[181] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, pages 1163–1171. 2016.

[182] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29:1163–1171, 2016.

[183] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv*, 2018.

[184] Shreyas Saxena, Oncel Tuzel, and Dennis DeCoste. Data parameters: A new family of parameters for learning a differentiable curriculum. In *Advances in Neural Information Processing Systems*, pages 11095–11105, 2019.

[185] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[186] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *CAIDA*, pages 309–318, 2001.

[187] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[188] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin, Madison, 2010.

[189] Shai Shalev-Shwartz and Yonatan Wexler. Minimizing the maximal loss: How and why. In *International Conference on Machine Learning (ICML)*, pages 793–801, 2016.

[190] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

[191] Saurabh Singh, Derek Hoiem, and David Forsyth. Swapout: Learning an ensemble of deep architectures. In *Advances in Neural Information Processing Systems (NIPS)*, pages 28–36. 2016.

[192] L. N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017.

[193] L. N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017.

[194] Michael R. Smith and Tony Martinez. A comparative evaluation of curriculum learning with filtering and boosting in supervised classification problems. *Comput. Intell.*, 32(2):167–195, 2016.

[195] Michael R. Smith, Tony Martinez, and Christophe Giraud-Carrier. An instance level analysis of data complexity. *Machine Learning*, 95(2):225–256, 2014.

[196] William G. Spady. Outcome-based education: Critical issues and answers. Technical report, American Association of School Administrators, 1994.

[197] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Baby Steps: How "Less is More" in unsupervised dependency parsing. In *NeurIPS 2009 Workshop on Grammar Induction, Representation of Language and Language Learning*, 2009.

[198] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014.

[199] Weijie Su, Stephen Boyd, and Emmanuel J. Candès. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. *Journal of Machine Learning Research*, 17(153):1–43, 2016.

[200] Amarnag Subramanya and Jeff Bilmes. Semi-supervised learning with measure propagation. *Journal of Machine Learning Research*, 12(Nov):3311–3370, 2011.

[201] James Steven Supancic III and Deva Ramanan. Self-paced learning for long-term tracking. In *CVPR*, pages 2379–2386, 2013.

[202] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[203] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. NIPS'99, page 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

[204] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Van-

houcke, and A. Rabinovich. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 00, pages 1–9, 2015.

[205] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[206] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018.

[207] Kevin Tang, Vignesh Ramanathan, Li Fei-fei, and Daphne Koller. Shifting weights: Adapting object detectors from image to video. In *NeurIPS*, pages 638–646, 2012.

[208] Ye Tang, Yu-Bin Yang, and Yang Gao. Self-paced dictionary learning for image classification. In *MM*, pages 833–836, 2012.

[209] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 1195–1204. 2017.

[210] William R Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3-4):285–294, 1933.

[211] Sunil Thulasidasan, Tanmoy Bhattacharya, Jeff Bilmes, Gopinath Chennupati, and Jamal Mohd-Yusof. Combating label noise in deep learning using abstention. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6234–6243, 2019.

[212] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[213] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018.

[214] Trieu H. Trinh, Minh-Thang Luong, and Quoc V. Le. Selfie: Self-supervised pretraining for image embedding, 2019.

[215] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5596–5605, 2017.

[216] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

[217] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[218] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 839–847, 2017.

[219] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 550–558. 2016.

[220] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[221] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning (ICML)*, volume 97, pages 6438–6447, 2019.

[222] Vikas Verma, Meng Qu, Alex Lamb, Yoshua Bengio, Juho Kannala, and Jian Tang. Graphmix: Regularized training of graph neural networks for semi-supervised learning. *ArXiv*, abs/1909.11715, 2019.

[223] Shengjie Wang, Wenruo Bai, Chandrashekhar Lavania, and Jeff Bilmes. Fixing mini-batch sequences with hierarchical robust partitioning. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 3352–3361, 2019.

[224] Shengjie Wang, Tianyi Zhou, and Jeff A. Bilmes. Bias also matters: Bias attribution for deep neural network explanation. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 6659–6667, 2019.

[225] Shengjie Wang, Tianyi Zhou, and Jeff A. Bilmes. Jumpout : Improved dropout for deep neural networks with ReLUs. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 6668–6676, 2019.

[226] Shengjie Wang, Tianyi Zhou, Chandrashekhar Lavania, and Jeff Bilmes. Constrained robust submodular partitioning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[227] Shengjie Wang, Tianyi Zhou, Chandrashekhar Lavania, and Jeff A. Bilmes. Constrained robust submodular partition. In *NeurIPS*, 2021.

[228] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 322–330, 2019.

[229] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. In *ICML*, 2014.

[230] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *ICML*, 2015.

[231] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris D. Bartels, and Jeff A. Bilmes. Submodular subset selection for large-scale speech training data. In *IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP) 2014*, pages 3311–3315, 2014.

[232] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[233] Andre Wibisono, Ashia C. Wilson, and Michael I. Jordan. A variational perspective on accelerated methods in optimization. *Proceedings of the National Academy of Sciences*, 113(47):E7351–E7358, 2016.

[234] Grant Wiggins and Jay. McTighe. Backward design. In *Understanding by Design*, pages 13–34. 1998.

[235] K. Woods, W. P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

[236] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017.

[237] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699, 2015.

[238] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[239] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7017–7025, 2019.

[240] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[241] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, number PART 1, pages 818–833, 2014.

[242] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[243] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[244] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[245] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

[246] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788, 2018.

[247] Denny Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16 (NeurIPS 2003)*, 2003.

[248] Tianyi Zhou and Jeff Bilmes. Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *ICLR*, 2018.

[249] Tianyi Zhou, Hua Ouyang, Jeff Bilmes, Yi Chang, and Carlos Guestrin. Scaling submodular maximization via pruned submodularity graphs. In *AISTATS*, 2017.

[250] Tianyi Zhou, Hua Ouyang, Jeff A. Bilmes, Yi Chang, and Carlos Guestrin. Scaling Submodular Maximization via Pruned Submodularity Graphs. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pages 316–324, 2017.

[251] Tianyi Zhou, Shengjie Wang, and Jeff Bilmes. Time-consistent self-supervision for semi-supervised learning. In *International Conference on Machine Learning (ICML)*, 2020.

[252] Tianyi Zhou, Shengjie Wang, and Jeff A Bilmes. Diverse ensemble evolution: Curriculum data-model marriage. In *Advances in Neural Information Processing Systems 31*, pages 5905–5916. 2018.

[253] Tianyi Zhou, Shengjie Wang, and Jeff A. Bilmes. Curriculum learning by dynamic instance hardness. In *NeurIPS*, 2020.

[254] Tianyi Zhou, Shengjie Wang, and Jeff A. Bilmes. Time-consistent self-supervision for semi-

supervised learning. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 11523–11533, 2020.

[255] Tianyi Zhou, Shengjie Wang, and Jeff A. Bilmes. Curriculum learning by optimizing learning dynamics. In *AISTATS*, 2021.

[256] Tianyi Zhou, Shengjie Wang, and Jeff A. Bilmes. Robust curriculum learning: From clean label detection to noisy label self-correction. In *ICLR*, 2021.

[257] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1):239–263, 2002.

[258] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[259] Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, pages 4083–4087, 2015.

[260] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002.

[261] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*, page 912–919, 2003.

[262] Xingquan Zhu, Xindong Wu, and Ying Yang. Dynamic classifier selection for effective mining from noisy data streams. In *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, pages 305–312, 2004.