Deep Learning with Abstention: Algorithms for Robust Training and Predictive Uncertainty

Sunil Thulasidasan

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

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Jeff Bilmes, Chair

Maryam Fazel

Sreeram Kannan

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Deep Learning with Abstention: Algorithms for Robust Training and Predictive Uncertainty

Sunil Thulasidasan

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

Machine learning using deep neural networks – also called "Deep Learning" – has been at the center of practically every major advance in artificial intelligence over the last several years, revolutionizing the fields of computer vision, speech recognition and machine translation. Spurred by these successes, there is now tremendous interest in using deep learning-based systems in all domains where computers can be used to make predictions after being trained on large quantities of data.

In this thesis, we tackle two important practical challenges that arise when using deep neural networks (DNNs) for classification. *First, we study the problem of how to robustly train deep models when the training data itself is unreliable.* This is a common occurrence in real-world deep learning where large quantities of data can be easily collected, but due to the enormous size of the datasets required for deep learning, perfectly labeling them is usually infeasible; when such label noise is significant, the performance of the resulting classifier can be severely affected. To tackle this, we devise a novel algorithmic framework for training deep models using an *abstention* based approach. We show how such a "deep abstaining classifier" can improve robustness to different types of label noise, and unlike other existing approaches, also *learns features* that are indicative of unreliable labels. State-of-the-art performance is achieved for noise-robust learning on a number of standard benchmarks; further gains

on noise-robustness are then shown by combining abstention with techniques from classical control and semi-supervised learning.

The second challenge we consider is improving the predictive uncertainty of DNNs. In many applications, and especially in high-risk ones, it is critical to have a reliable measure of confidence in the model's prediction. DNNs, however, often exhibit pathological *overconfidence*, rendering standard methods of confidence-based thresholding ineffective. We first take a closer look at the roots of overconfidence in DNNs, discovering that introducing uncertainty into the training labels leads to significantly reduced overconfidence and improved probability estimates associated with predicted outcomes; using this observation, we show how threshold-based abstention can be made to work again. Then we consider the problem of open set detection – where the classifier is presented with an instance from an unknown category – and demonstrate how our abstention framework can be a very reliable open set detector.

The contributions of this thesis are thus two-fold, and in two parts – part one on robust training, and part two on predictive uncertainty – but unified by a common theme: abstention. Our work demonstrates that such an approach is highly effective both while training and deploying DNNs for classification, and hence, a useful addition to a real-world deep learning pipeline.

TABLE OF CONTENTS

Page

List of Figures iv

List of Tables ix

Glossary xi

Chapter 1: **Introduction** 1
- 1.1 Motivation 2
- 1.2 Contributions 4
- 1.3 How this Thesis is Organized 5

Part I: **Robust Training of Deep Neural Networks** . . 7

Chapter 2: **Label Noise in Deep Learning** 8
- 2.1 An Overview of Label Noise 8
- 2.2 Preliminaries 10
- 2.3 Learning in the Presence of Label Noise: Problem Setting and Taxonomy . . 13
- 2.4 Intrinsic Noise Robustness 15
- 2.5 Noise-Aware Approaches 17
- 2.6 Noise Filtering 22
- 2.7 Conclusion 27

Chapter 3: **Combating Label Noise Using Abstention** 28
- 3.1 Introduction: The Deep Abstaining Classifier 28
- 3.2 Loss Function for the DAC 30
- 3.3 The DAC as a Learner of Structured Noise 36
- 3.4 Learning in the Presence of Unstructured Noise: The DAC as a Data Cleaner 46
- 3.5 Abstention and Memorization 51

3.6 Conclusions . . . 52

Chapter 4: **Extensions to the Deep Abstaining Classifier** . . . 53
4.1 Stabilizing Abstention Behavior of the DAC . . . 53
4.2 An Overview of PID Control . . . 55
4.3 Stabilizing Abstention with PID Control . . . 57
4.4 Experiments . . . 59
4.5 Setting the Setpoint: Estimating Label Noise via Sampling . . . 63
4.6 Discussion . . . 67
4.7 Semi-Supervised Deep Learning with the Deep Abstaining Classifier . . . 67
4.8 Improving Label-Noise Robustness with the DAC and SSL: A Two-Step Approach 76
4.9 Conclusion . . . 79

Part II: **Improving Predictive Uncertainty in Deep Learning** . . . 80

Chapter 5: **Uncertainty in Deep Learning: An Overview** . . . 81
5.1 Introduction . . . 81
5.2 Classification with Rejection . . . 83
5.3 Overconfidence and Miscalibration in Deep Learning . . . 87
5.4 A Closer Look at Softmax . . . 91
5.5 Bayesian Deep Learning . . . 94

Chapter 6: **Making Threshold-based Abstention Work: Improving Calibration in Deep Neural Networks** . . . 97
6.1 Are DNNs Trained to be Overconfident? . . . 97
6.2 An Overview of Mixup Training . . . 99
6.3 Experiments . . . 101
6.4 Effect of Soft Labels on Calibration . . . 111
6.5 Testing on Out-of-Distribution and Random Data . . . 113
6.6 Conclusion . . . 117

Chapter 7: **Knows When it Doesn't Know: Open Set and Out-of-Distribution Detection Using Abstention** . . . 118
7.1 Background . . . 118

7.2 Related Work 119
7.3 Out-of-Distribution Detection with the DAC 125
7.4 Experiments 129
7.5 Conclusion 139

Chapter 8: **Future Directions** 140

Bibliography 143

LIST OF FIGURES

Figure Number Page

2.1 A constrained linear layer inserted between softmax layer and loss, with the weight matrix of this layer modeling noise. Output probabilities are transformed from the base model into a distribution that better matches the noisy labels. Figure reproduced from (Sukhbaatar et al., 2014) 18

2.2 Training error decreases consistently but the validatin error eventually increases, suggesting that DNN is overfitting to the traiing data. Figure reproduced from (Goodfellow et al., 2016) 23

2.3 DNNs learn differently on clean and mislabeled data. (a) Train loss evolution on clean and noisy samples on CIFAR-10 under 80% label noise. Reproduced from (Arazo et al., 2019). (b) Loss histograms on clean and noisy data for 40% label noise on CIFAR-100 at train accuracy of 50% for symmetric noise and, (c) for pair noise. Reproduced from (Song et al., 2019). 24

3.1 DAC behavior (where 10% of the data has corrupted labels) for different fixed α's (indicated by color) illustrating the tendency for all-or-nothing abstention if α is kept constant. 35

3.2 (a) A sample of smudged images on which labels were randomized.(b) Abstention percentage on training set as training progresses (c)The DAC abstains on most of the smudged images on the test set (abstention recall) (d) Almost all of the abstained images were those that were smudged (abstention precision) (e) Risk-coverage curves for baseline DNNs, DAC and post-DAC DNN for both softmax-based thresholds and SGR. First-pass training with the DAC improves performance for both softmax and SGR methods. 39

3.3 (a) All monkey images in the trainset had their labels randomized (b) The DAC abstains on about 80% of the monkey images (abstention precision). (c) Among images that were abstained, most of the images were those of monkeys (abstention precision). (d) Distribution of baseline DNN predictions over monkey images in the test set indicating learning difficulty on this class (e) Distribution of winning softmax scores of the baseline DNN on the monkey images (f) Distribution of baseline DNN softmax scores > 0.9. Most of these confident predictions are non-monkeys (g) Comparison against various abstention methods (softmax and SGR) on all the test images (h) Same comparison, but only on those images on which the DAC abstained. The DAC has a small but consistent advantage in both cases. All figures are computed on the test set. 42

3.4 Results on blurred-image experiment with noisy labels (a)20% of the images are blurred in the train set, and their labels randomized (b) Validation accuracy for baseline vs DAC (non-abstained) (c)Abstention behavior for the DAC during training (d) Distribution of predictions on the blurred validation images for the DAC. We also observed (not shown) that for the baseline DNN, the accuracy on the blurred images in the validation set is no better than random. 44

3.5 Filter visualizations for the DAC. When presented with a smudged image (a,c), the smudge completely dominates the feature saliency map (b,d) that cause the DAC to abstain. However for the same image without a smudge (e), the class features become much more salient (f) resulting in a correct prediction. For abstention on monkeys (g), the monkey features are picked up correctly (h), which leads to abstention 45

4.1 Abstention on train set at various levels of label randomization (indicated by color). When α (not shown), is linearly ramped up, the DAC starts overfitting and abstention eventually decays to zero. 55

4.2 Performance of a PID-controlled deep abstaining classifier on various datasets with structured noise, in terms of abstention stablization to set-point (dotted red line) and the quality of abstention (precision and recall). 60

4.3 Performance of a PID-controlled deep abstaining classifier on the MNIST dataset on ResNet-18 architecture, at various levels of label noise. Shown are abstention stabilization to set-point and the quality of abstention (precision and recall). 61

4.4 Performance of a PID-controlled deep abstaining classifier on the CIFAR-10 and CIFAR-100 on two different architectures, at various levels of label noise. Shown are abstention stabilization to set-point and the quality of abstention (precision and recall). 62

4.5 Estimated label noise at various sample sizes and at different noise rates on various image datasets. MNIST, STL-10 and CIFAR-10 have objects from ten different categories, while CIFAR-100 consists of one hundred categories. . . 66

4.6 Data are assumed to lie on a low-dimensional manifold with nearby points belonging to the same class 69

4.7 A graph can be induced on the low-dimensional manifold. In graph-based SSL, the graph structure is used to enforce the smoothness assumption. 70

4.8 Two-stage training with the DAC. In Stage one, the DAC is used to identify noisily labeled training data through abstention. In Stage two, all training data that were abstained on are treated as unlabeled data. The labeled and unlabeled data are then used together for semi-supervised learning. 77

5.1 An illustration of likelihood-based classification 83

5.2 Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Figure reproduced from (Guo et al., 2017) 88

5.3 Accuracy vs confidence (captured by mean of the winning score) on the CIFAR-100 validation set at different training epochs for the VGG-16 deep neural network. The DNN moves from underconfidence, at the beginning of training, to overconfidence at the end. A well-calibrated classifier would have most of the density lying on the $x = y$ gray line 89

5.4 A sample of images where a DNN makes very high confident predictions. In this case, the predictions were "starfish" (a),"king penguin" (b) and "cat"(c), all with softmax confidence > 0.99). Images (a) and (b) reproduced from `http://www.evolvingai.org/fooling` 90

6.1 Joint density plots of accuracy vs confidence (captured by the mean of the winning softmax score) on the CIFAR-100 validation set at different training epochs for the VGG-16 deep neural network. **Top Row:** In regular training, the DNN moves from under-confidence, at the beginning of training, to overconfidence at the end. A well-calibrated classifier would have most of the density lying on the $x = y$ gray line. **Bottom Row:** Training with mixup on the same architecture and dataset. At corresponding epochs, the network is much better calibrated. 98

6.2 Calibration results for mixup and baseline (no mixup) on various image datasets and architectures. **Top Row**: Scatterplots for accuracy and confidence for STL-10(a,b) and CIFAR-100(c,d). The mixup case is much better calibrated with the points lying closer to the $x = y$ line, while in the baseline, points tend to lie in the overconfident region. **Middle Row:** Mixup versus comparison methods where label_smoothing is the ϵ-label smoothing method and ERL is the entropy regularized loss. **Bottom Row**: Expected calibration error (e) and overconfidence error (f) on various architectures. Experiments suggest best ECE is achieved for α in the [0.2,0.4] (h), while overconfidence error decreases monotonically with α due to under-fitting (i). Accuracy behavior for differently calibrated models is shown in (j). 104

6.3 Calibration on ImageNet for ResNet architectures 105

6.4 Accuracy, calibration and overconfidence on various NLP datasets 106

6.5 Distribution of winning scores on various image datasets 108

6.6 Training loss and calibration error under extended training for CIFAR-10 and CIFAR-100 with mixup. Baseline (no mixup) training loss (orange) goes to zero early on while mixup continues to have non-zero training loss even after 1000 epochs. Meanwhile, calibration error for mixup does not exhibit an upward trend even after extended training. 110

6.7 Entropy distribution of training labels as a function of the α parameter of the $Beta(\alpha, \alpha)$ distribution from which the mixing parameter is sampled. 112

6.8 Calibration performance when only features are mixed vs. full mixup and other baselines on various datasets and architectures 113

6.9 Distribution of winning scores from various models when tested on out-of-distribution and gaussian noise samples, after being trained on the STL-10 dataset. 114

6.10 Prediction behavior as one moves away from the training data 116

7.1 An illustration of the separability of scores on in and out-of-distribution data for a regular DNN (left) and the Deep Abstaining Classifier (DAC) (Right); the presence of the abstention class enables the DAC to produce higher separation and thus, better detection, on OoD samples. 127

7.2 Images and filter visualizations illustrating the usage of DAC as an out-of-category detector. During inference, all incoming images are augmented with a fixed feature (a smudge for instance, shown as a blue streak here.) When presented with known classes, (horse (a) or car (c)), the activations for the class features are much more salient than the smudge (b,d). For objects from unknown classes (e,g,i), the absence of known features causes the smudge to be the most salient feature (f,h,j) resulting in abstention. 128

LIST OF TABLES

Table Number Page

3.1 Comparison of performance of DAC against related work for data corrupted with uniform label-noise. The DAC is used to first filter out noisy samples from the training set and a DNN is then trained on the cleaner set. Each set also shows the performance of the baseline DNN trained without removing noisy data. Also shown is the performance of a hypothetical oracle data-cleaner that has perfect information about noisy labels. The numbers in parentheses next to the DAC indicate the fraction of training data filtered out by the DAC and the remaining noise level. For the oracle, we report just the amount of filtered data (remaining noise level being zero). $\mathcal{L}_q$ and truncated $\mathcal{L}_q$ results reproduced from (Zhang and Sabuncu, 2018); MentorNet results reproduced from (Jiang et al., 2018b) 48

3.2 Comparison of DAC vs related methods for class-dependent label noise. Performance numbers reproduced from (Zhang and Sabuncu, 2018). For the DAC, an abstaining classifier is first used to identify and eliminate label noise, and an identical DNN is then used for downstream training. 50

4.1 Results for abstention training followed by semi-supervised training using the MixMatch algorithm. The DAC is trained on different datasets and architectures. Downstream SSL training is always done on the WideResNet 28x10 architecture. DAC and Baseline results reproduced from Chapter 3 . . 79

6.1 Mixup results on CIFAR datasets with the ResNet 18 architecture 108

6.2 Manifold mixup experiments. For the extended training experiments, we use the same setup as (Verma et al., 2018) while for the experiments that trained for 200 epochs, we anneal the learning rate at epoch 60, 120 and 160 while keeping all other hyperparameters fixed to match the regular mixup experiments in Section 6.3 111

6.3 Out-of-category detection results for the DAC on STL-10 and Tiny ImageNet. 115

7.1 Comparison of the extra class method (ours) with various other out-of-distribution detection methods when trained on CIFAR-10 and CIFAR-100 and tested on other datasets. All numbers from comparison methods are sourced from their respective original publications. For our method, we also report the standard deviation over five runs (indicated by the subscript), and treat the performance of other methods within one standard deviations as equivalent to ours. For fair comparison with the Mahalanobis detector (MAH) (Lee et al., 2018), we use results when their method was not tuned separately on each OoD test set (Table 6 in (Lee et al., 2018). 133

7.2 DAC vs OpenMax. The OpenMax implementation was based on code available at `https://github.com/abhijitbendale/OSDN` and re-implemented by us in PyTorch (Paszke et al., 2019). 134

7.3 OoD detection performance of DAC compared to approaches that model uncertainty in deep learning. The performance of DAC as an OoD detector, evaluated on various metrics and compared against competing baselines. ↑ and ↓ indicate that higher and lower values are better, respectively. Best performing methods (ignoring statistically insignificant differences)on each metric are in bold. All methods were re-implemented using open-sourced code where available. 138

GLOSSARY

CIFAR-10/100: Image dataset consisting of 60,000 images of 32×32 pixel resolution. Labels are over 10 broad categories (CIFAR-10) or 100 fine categories (CIFAR-100). Frequently used as a benchmark for image classifiers.

CCE: Categorical Cross Entropy

DAC: Deep Abstaining Classifier

DNN: Deep Neural Network

EM: Expectation-Maximization

ERM: Empirical Risk Minimization

MNIST: Hand-written digit dataset, of 60,000 total examples of digits 0 through 9

PID CONTROLLER: Proportional-Integral-Derivative Controller

OoD: Out-of-Distribution

OSR: Open Set Recognition

SGD: Stochastic Gradient Descent

SSL: Semi-Supervised Learning

STL-10: Image dataset consisting of 5000 labeled training images, and 8000 test images, over 10 broad categories. 96×96 resolution. Frequently used as a benchmark for image classifiers.

SVM: Support Vector Machine

ACKNOWLEDGMENTS

This dissertation is the culmination of my doctoral studies at the University of Washington, most of which were conducted while working as a full-time scientist at the Los Alamos National Laboratory. I had already been working at LANL for a number of years when, in 2014, I decided I was ready for further intellectual challenges and started my PhD journey in this new and exciting field called deep learning. It has been an immensely rewarding six-year journey, full of intellectual lessons and life lessons, and as I wrap up this chapter of my life, I am grateful to LANL, to UW ECE and to the numerous people who helped me succeed in this undertaking.

I would like to start by sincerely thanking my advisor Jeff Bilmes for his guidance and support, and his willingness to accommodate my non-traditional arrangement as a long-distance PhD student. Being able to remotely participate on a regular basis in group meetings with Jeff and other members of Melodi lab over the years helped me stay engaged and lessened the isolation of being far removed from campus life. Jeff has also been very accommodating of my research interests and I have always benefited from his rigor, crucial insights and critical feedback during our interactions.

I would also like to express my sincerest thanks to Professors Maryam Fazel, Sreeram Kannan and Archis Ghate, who graciously agreed to serve on my committee and provided valuable feedback on my research.

I spent the 2014-2015 academic year in Seattle; taking a year of absence from work to pursue academic goals without having to worry about future employment would have been hardly possible without supportive colleagues and managers. In particular, I would like to express my sincere gratitude to Stephan Eidenbenz, Patrick Kelly and Stephen Lee for their

willingness to allow me to make that career pivot, and to Amy Larson and Garrett Kenyon for their support during the early years. I would also like to thank my former mentor Madhav Marathe and Professor Randall LeVeque at UW for their willingness to vouch for my return to graduate school.

I've been very fortunate that my PhD research has been well aligned with my work at LANL, and for this, many thanks are owed to Tanmoy Bhattacharya, whom I've had the pleasure of working with as I developed the ideas described in this thesis, as well as to my colleague, friend and enthusiastic collaborator Gopinath Chennupati.

Deep learning is a computationally intensive science, and thus I'm immensely thankful for the plentiful computer resources that were available to me at LANL, and to Dave Rich and team for maintaining a state-of-the-art cluster that allowed me to crank out the numerous experimental results that were required for this effort.

At UW, special thanks go out to Brenda Larson of ECE for helping me navigate the procedural side of graduate school, always responsive and always patient, especially during the final months of my PhD which were completed amid the worldwide COVID-19 pandemic lockdown. I would also like to thank current and former Melodi lab students: Chandrashekhar Lavania, Neeraja Abhyankar, Tianyi Zhou, Shengjie Wang, Lilly Kumari and Wenruo Bai — it has been a pleasure getting to know all of you and your research. The year that I spent in Seattle was also made immensely more enjoyable due to the numerous friendships that I forged with fellow students – thank you Noshad Bagha, Vijeth Rai, Yuguang Li, Marcos Inonan, Qiuyu Chen and Astrini Sie for good times and fond memories.

Holding down a full-time job and being a parent to two young children while pursuing a PhD was only made possible with the support of wonderful family members. I would like to thank my father for encouraging me to embark on a PhD — a reflection of his own love of lifelong learning — and my mother and my sister Preetha for cheering me on. I would also like to thank my parents and my mother-in-law Lata for their help during the time I was

away in Seattle. And of course, absolutely none of this would have been possible without my wife Namrata's unwavering support from start to end — thank you for your love, for your patience and thank you for believing in me.

And finally, this journey has been made immeasurably more joyful by being able to share it with my children Dhruv and Reya — thank you for the little gifts, the numerous foosball games, many sweet words of encouragement and constantly checking on my progress. It has been a sheer blessing to have you both in my life.

DEDICATION

To my parents, Thulasidasan Pillai and Chandramathy Thulasidasan.

To Dhruv and Reya.

To Namrata.

Chapter 1

INTRODUCTION

Machine learning enables computers to learn from data, and is now the dominant paradigm in artificial intelligence (AI). Instead of purpose-built programs that use hand-coded rules, machine learning algorithms instead use statistics and a large number of training examples to teach computers how to accurately recognize patterns in data (Murphy, 2012; Hastie et al., 2009; Bishop, 2006).

The last decade has been an especially fruitful one for machine learning-based AI, witnessing dramatic improvements in the ability of computers to perform perceptual tasks such as object detection and speech recognition. Central to these advancements has been a particular type of machine learning known as *Deep Learning* (LeCun et al., 2015): by utilizing a very large number of individual computational units, called *neurons*, connected in a layer-wise fashion – an architecture vaguely inspired by the human brain – deep learning enables computers to learn about the world using a hierarchy of concepts (Goodfellow et al., 2016). When trained on millions of examples, the performance improvements compared to earlier methods have been so striking that this approach has now revolutionized the fields of computer vision, automatic speech recognition and language translation (Lewis-Kraus, 2016)

Inspired by these successes, there is now tremendous interest and enthusiasm for using deep learning in all domains where large amounts of data are available for training computers to make predictions. Indeed, deep learning is the main focus of most of the current investment in AI (Marcus and Davis, 2019) and is now being deployed – or at least being considered – in fields such as medical diagnostics(Litjens et al., 2017), autonomous driving(Falcini et al., 2017; Huval et al., 2015) and even in legal practice(Chalkidis and Kampas, 2019).

1.1 Motivation

The work in this thesis is motivated by practical challenges that arise when using deep learning for *classification*, an area where it has had the most impact (Goodfellow et al., 2016, p.22). In classification, the goal is to correctly identify which category a given data sample belongs to, such as recognizing a person in a photograph, or categorizing the type of disease based on a patient's chest X-ray. In machine learning, this is achieved by training the classifier on examples and their correct labels from each category; by seeing numerous such examples, the model learns to associate the features of a category with the correct label.

Deep neural networks (DNNs) excel at this kind of learning, also known as *supervised learning* (because the learning is guided by the presence of the correct label); when trained in this fashion, state-of-the-art deep models have been able to even surpass the ability of experienced human specialists(Nam et al., 2019). But to achieve this performance, a DNN has to be trained on a *very large number of correctly labeled training samples*. For example, the ImageNet dataset (Deng et al., 2009), which played a critical role in establishing the dominance of deep learning in computer vision (Krizhevsky et al., 2012), originally consisted of 3.2 million images, collected from the Internet. While the data collection was relatively straightforward, correctly labeling them took almost *three years* of coordinated human effort (Fei-Fei and Deng, 2017).

In most real-life situations, however, this kind of effort is simply infeasible at the scale required for deep learning. It is often easier to acquire training labels for a large dataset either by using an already-trained weaker model or using "side information" to automatically guess the labels. In either case, the resulting dataset, while large, is likely to have labeling errors – a phenomenon referred to as *label noise*. When such label noise is significant – a common scenario with real-world data – the performance of deep models can be severely degraded (Sukhbaatar et al., 2014; Patrini et al., 2017; Zhang et al., 2016).

Thus the **first challenge** we tackle in this thesis is: *how do we robustly train deep learning models for classification in the presence of label noise, even when the label noise can*

occur at significant amounts?

Owing to their success in perceptual tasks like vision, DNNs are also being considered in domains like medical diagnosis (Miotto et al., 2016; Litjens et al., 2017) where it is not only important to make accurate predictions, but also critical to have a *reliable measure of confidence* in the model's prediction; when the model is *uncertain*, it should refrain from making a prediction and route the decision to a human expert.

The learning capacity of DNNs, however, also comes with unique challenges for predictive uncertainty. When presented with objects from categories not seen during training – known as "out-of-distribution" or "open world" (Bendale and Boult, 2016) settings[1] – or *even when presented with random noise*, DNNs often exhibit pathological *overconfidence* (Nguyen et al., 2015). Standard methods of confidence-based thresholding are simply ineffective in such situations while full Bayesian approaches, that incorporate model uncertainty, are impractical owing to the vast number of parameters that make up a DNN(Wang and Yeung, 2016).

Further, even on categories seen during training (in-distribution data), the predictions of DNNs suffer from poorly calibrated confidence. In a well-calibrated classifier, probabilistic predictive scores should be indicative of the actual likelihood of correctness. However, due to their capacity to overfit, modern DNNs tend to be miscalibrated (Guo et al., 2017), and furthermore, we find that proposed remedies, while improving calibration on known categories, end up reducing the ability of the DNN to discriminate between known and unknown categories.

Hence, the **second challenge** we consider in this thesis is: *how do we improve the predictive uncertainty and calibration of deep models when test data can come from both known and unknown categories, and do so in a computationally practical way?*

We note that adversarial settings (Goodfellow et al., 2014b) – where the input has been maliciously perturbed in a very specific way to cause the DNN to output a wrong prediction – are not considered here. While, this is a very active area of research in deep learning (Akhtar

[1]These terms are used interchangeably in the literature

and Mian, 2018; Papernot, 2018; Hosseini, 2019) it is outside the scope of the current work. We restrict our study to non-adversarial scenarios, since a vast number – and perhaps the majority – of present-day applications for deep learning are in this setting.

1.2 Contributions

The main contribution of this thesis is *a novel framework to tackle the challenges of both robust training and predictive uncertainty in deep learning* using the principle of *abstention*. This allows the model to *defer learning* on potentially noisy data, and *refrain from predicting* in uncertain scenarios. In machine learning, abstention has so far only been applied during the prediction stage (Chow, 1970; De Stefano et al., 2000; Hellman, 1970; Fumera and Roli, 2002; Cortes et al., 2016; Geifman and El-Yaniv, 2017); our work is the *first* to demonstrate its usefulness even during training. Concretely, the contributions in this work are:

- We devise a novel way to train a DNN using an *abstention*-based loss function. We show both analytically and empirically how such a "deep abstaining classifier" (DAC) allows a DNN to abstain on confusing samples while continuing to learn and improve classification performance on the non-abstained samples. We empirically demonstrate how such an approach not only improves robustness to different types of label noise, but unlike other existing approaches, also *learns features* that are indicative of unreliable labels. We show state-of-the-art performance for noise-robust learning on a number of standard benchmarks.

- Next, we show how abstention training can be combined with classical control – specifically, PID control – to reliably identify mislabeled samples. This is one of the few works that use control techniques to tune the training of DNNs. Further, we show how the resulting abstaining classifier can be combined with modern semi-supervised learning to achieve robustness even in the presence of significant amounts of label noise.

- We take a closer look at the roots of overconfidence in DNNs, and show that the

calibration of modern neural networks is affected by standard methods of training. We show how label smoothing can have a surprisingly beneficial effect on confidence calibration, being the the first work to explore this effect; we leverage this to show how threshold-based abstention can be made to work again.

- Finally, we attack the open set detection problem in image classification – where a classifier is presented with an object from an unknown category – and show how the DAC can reliably abstain on predicting in such scenarios, outperforming other methods in the field.

The contributions of this thesis are thus two-fold: (i) robust training in the presence of noisy data, and (ii) improved predictive uncertainty in deep learning, but unified by the common theme of abstention. The results in this thesis demonstrate the utility of such an approach for real-world deep learning pipelines.

1.3 How this Thesis is Organized

The thesis is organized into two parts – part one devoted to robust training, and part two to predictive uncertainty, organized as follows:

In *Chapter 2*, we formally introduce the problem of label noise, its sources and consequences in deep learning. We go over some mathematical preliminaries, and then take a detailed look at various existing approaches, including very recent ones, for training deep models in the presence of label noise.

In *Chapter 3*, we introduce the abstention model, and the "Deep Abstaining Classifier" (DAC). We introduce the abstention loss function, analyze its properties and show how training using abstention achieves strong noise robustness under various label noise scenarios; state-of-the-art results are achieved in various settings. This chapter is based on our work that appeared in ICML 2019 (Thulasidasan et al., 2019a).

In *Chapter 4,* we describe extensions to the abstention training framework. The DAC is combined with PID control and then with semi-supervised learning to further demonstrate

performance gains for noise-robustness.

Part two begins in *Chapter* 5, where we examine predictive uncertainty in deep learning. We take a look at the current landscape in uncertainty modeling for deep learning, discuss the unique challenges that arise in deep learning and see how traditional threshold-based rejection approaches are often ineffective for such models.

We take a closer look at the issues of overconfidence and calibration in *Chapter* 6, exploring an aspect of DNN training that contributes to overconfident predictions. Our work is the first to explore this and we present compelling empirical evidence that shows how the standard approach of training with hard labels contributes to overconfidence in DNNs. Using this insight, we show how calibration and threshold-based abstention can be improved for deep learning; this chapter is based on our work that appeared in NeurIPS 2019 (Thulasidasan et al., 2019b)

And finally, in *Chapter* 7, we see how the abstention framework developed in part one can be used for open set and out-of-distribution detection for image classification, improving upon existing methods in the field.

We conclude in *Chapter* 8 briefly discussing various directions for extending the current line of work to other areas in deep learning.

Part I

ROBUST TRAINING OF DEEP NEURAL NETWORKS

Chapter 2

LABEL NOISE IN DEEP LEARNING

In this chapter, we formally introduce the problem of label noise in deep learning. We first briefly discuss the sources of label noise and the effects of learning in the presence of such noise. We then provide a detailed overview of various approaches to tackle label noise, focusing on the work in the deep learning domain.

2.1 An Overview of Label Noise

Deep learning is a data-hungry approach to machine learning, and while enabling deep models to match or even surpass human performance, the requirement of vast quantities of human-annotated datasets is a significant bottleneck in training of such models. To overcome this, generally one of two approaches are employed: labeling by enlisting the services of a large number of people by using a service like the Amazon Mechanical Turk (Rashtchian et al., 2010) or alternatively, automatically collecting large amounts of data from the Internet, that have associated meta-information (tags, for instance) which are then used as labels. In both cases, erroneously labeled data is bound to occur, often in significant fractions for the latter situation (Li et al., 2017a).

This phenomenon of mislabeled data – referred to as *label noise* – can be due to a number of reasons: for data automatically collected from the Internet, associated tags and other meta-information often only provide *weak labels*; keywords might only be weakly related to the correct object label or worse, might be completely irrelevant, further adding to label noise. In other cases, the required information to distinguish between similar categories might not be present in the data due to poor quality or the presence of confusing features. For human-labeled sets, sufficient expertise might not be available to reliably label a given sample,

especially in the case of crowdsourcing. Even in the relatively rare and fortunate situations when significant amounts of expertly-labeled data are available, when the labeling task is subjective, experts themselves might disagree on the correct label. While it is generally not necessary to model the source of label noise to mitigate its effect on training, in some cases knowing the source of label noise might help in identifying mislabeled data. A more detailed discussion on label noise in traditional machine learning this is given in (Frénay and Verleysen, 2014).

Label noise is thus an inescapable fact of real-world deep learning and hence, it is important to understand the consequences of learning in the presence of mislabeled data. Fundamentally, label noise is an inconsistent mapping between features and the label for a given category, obscuring their relationship (Frénay and Verleysen, 2014). While errors can also affect input features, the effect of label noise can be especially significant (Zhu and Wu, 2004). Intuitively, it is easy to see why: for any given sample, there are many features, but only one class[1]. Features might have varying levels of importance depending on the task, but the label is always important and has a large impact on learning.

Hence, it is not surprising that for many learning algorithms, labeling errors have been shown to degrade classifier performance (Zhu and Wu, 2004). Indeed, for low capacity classifiers, even mild class-independent noise has been shown to produce solutions that are little better than random guessing (Long and Servedio, 2010). For example, the k nearest neighbor classifier is especially susceptible to label noise when $k = 1$, and the optimal number of neighbors is shown to monotonically increase with increasing amounts of label noise (Okamoto and Yugami, 1997). Boosting (Freund et al., 1999) is another technique that is affected by label noise (McDonald et al., 2003; Melville et al., 2004), with the adaptive boosting (AdaBoost) being more severely affected since it tends to give more importance to mislabeled instances (Dietterich, 2000). Further, both sample and classifier complexity increases in the presence of label noise. For instance, using the "Probably Approximately

[1]We assume we are dealing single-label classification scenarios

Correct" (PAC) (Valiant, 1984) learning framework, it was shown in (Angluin and Laird, 1988) that the number of necessary samples required for learning in binary-class settings increases with label noise, while decision trees (Quinlan, 1986) were shown to increase in size when learning on noisy data (Brodley and Friedl, 1999).

Given the large size of training datasets in deep learning, small amounts of label noise are usually not an issue. But for large, weakly labeled data automatically collected from the Internet or from other sources, labeling errors can be as high as 40 to 50% (Li et al., 2017a; Xiao et al., 2015). The results in (Zhang et al., 2016) show that DNNs, owing to their large capacity, can easily fit all the mislabeled training data, resulting in significant performance degradation of the final classifier; this has also been observed in various other works (Sukhbaatar et al., 2014; Patrini et al., 2017). We cover these and other works, and approaches to tackling label noise in the later sections in this chapter.

2.2 Preliminaries

Before diving into the existing literature on label noise, we review some preliminary concepts and fix our mathematical notation, mostly following the notation in (Patrini et al., 2017) and (Ghosh et al., 2017). We assume we are working in a K-class classification scenario with feature space denoted by $\mathcal{X} \in \mathbb{R}^d$ and label space by $\mathcal{Y}$. If the labels are *hard labels*, then $\mathcal{Y} = \{y : y \in \{0,1\}^K, \mathbf{1}^T y = 1\}$. Alternatively, if the labels are *soft labels*, then $\mathcal{Y} = \{y : y \in [0,1]^K, \mathbf{1}^T y = 1\}$. Unless explicitly specified, we assume we are always working in the hard-label scenario. We use t_i to represent the given class label of x_i; that is, $t_i = \text{argmax}_j\, y_{i,j}$. A training sample is the ordered pair $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, drawn from an unknown distribution $p(x, y)$; we denote expectations over this joint distribution by $\mathbb{E}_{x,y}$.

We denote our classifier as $f : \mathcal{X} \to \triangle^{K-1}$, the $K-1$-dimensional probability simplex representing a discrete probability distribution over K classes. In the case of an n-layer DNN, f comprises a series of transformations, that is $f = s \circ h^{(n)} \circ h^{(n-1)} \circ \ldots h^{(1)}$, where the

intermediate layers $h^{(i)}$ are defined as

$$\begin{aligned} (\forall i \in [n-1]),\, h^{(i)}(z) &= \sigma\left(W^{(i)}z + b^{(i)}\right) \\ h^{(n)}(z) &= W^{(i)}z + b^{(i)} \end{aligned} \tag{2.1}$$

with $W^{(i)} \in \mathbb{R}^{d^{(i)} \times d^{(i-1)}}$ and $b^{(i)} \in \mathbb{R}^{d^{(i)}}$ being the parameters to be estimated, and σ is any element-wise activation function. A commonly used function is the Rectified Linear Unit (ReLU) (Zeiler et al., 2013), in which case, $\sigma(x) = \max(0, x_i)$. For ease of notation, we denote the entire set of learnable parameters $\{W^{(1)}, \ldots W^{(n)}, b^{(1)}, \ldots b^{(n)}\}$ by Θ.

Since we're training a K-way classifier, the output $h(.)$ is converted to a probability distribution over the K classes by passing it through a *softmax* layer s, where

$$s_i(z) = \frac{\exp(h_i(z))}{\sum_{k=1}^{K} \exp(h_k(z))} \tag{2.2}$$

The output $f(\Theta, x)$ can be viewed as the DNN's estimate of the class conditional probabilities $p(y|x)$; often we will omit Θ, and just denote the output by $f(x)$.

To train a classifier, we use a *loss function* $\mathcal{L}$ to measure the discrepancy between the predicted and given labels. For any given loss function $\mathcal{L}$, a classifier f, and training inputs (x, y), we define the $\mathcal{L}-$risk of f by

$$R_{\mathcal{L}}(f) = \mathbb{E}_{x,y}\left[\mathcal{L}\left(f(x), y\right)\right] \tag{2.3}$$

Since we do not know the true distribution $p(x, y)$, we compute an approximation to the $R_{\mathcal{L}}$, called the *empirical* $\mathcal{L}-$risk, defined as

$$R_{\mathcal{L}}^{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(f(x_i), y_i\right) \tag{2.4}$$

and then train using the principle of *Empirical Risk Minimization*, that is we learn a classifier f that minimizes $R_{\mathcal{L}}^{emp}$. Note that $\mathcal{L}-$risk depends on $\mathcal{L}$, our loss function. One

choice for $\mathcal{L}$ could simply be the measure the number of prediction errors as follows: denoting the predicted label for a sample x_i as $\hat{t}_i = \text{argmax}_j\, f_j(x_i)$, one could use use a $0-1$ loss defined as $\mathcal{L}(t, \hat{t}) = \mathbf{1}(t \neq \hat{t})$. However, the $0-1$ loss is not differentiable, and thus hard to directly use in machine learning, where optimization is often performed using gradient descent.

To overcome this, we use a *surrogate loss* (Bartlett et al., 2006) which is used as a proxy for the loss we actually want to optimize, and which unlike the original loss, can be optimized efficiently. The use of surrogate losses are common in many machine learning settings, and for the above K-class scenario, a standard surrogate loss is the *categorical cross entropy* loss, defined on a sample x as:

$$\mathcal{L}(f(\Theta, x)) = -\sum_{j=1}^{K} y_i \log(f_i(\Theta, x)) \tag{2.5}$$

where $y \in 0, 1^K$ is the given hard label of x. cross entropy, being differentiable, can be minimized via gradient descent. For DNNs, the standard way to perform this minimization is via *Stochastic* Gradient Descent (SGD)(Bottou, 2010). First, the loss over the entire training set is approximated by computing the loss over a *mini-batch* $\mathcal{B}$ of samples:

$$\mathcal{L}_{\mathcal{B}}(f(\Theta)) = \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \mathcal{L}\left(f(x_i, \Theta), y_i)\right), \tag{2.6}$$

followed by the parameter update step using gradient descent, given by

$$\Theta \leftarrow \Theta - \mu \nabla_{\Theta} \mathcal{L}_{\mathcal{B}}(f) \tag{2.7}$$

where μ is the learning rate. The idea behind SGD is that since gradient is an expectation, it can be approximately estimated using a small set of samples, which makes optimization feasible when dealing with large amounts of data. Further, even though, in deep learning, the loss function is nonconvex in the parameters Θ and SGD is not guaranteed to find a global

minimum, in practice it works very well; indeed SGD is the workhorse algorithm in all of modern deep learning.

Unlike classical optimization, in machine learning what is of interest is the classification error on data that have *not been used* for training (referred to as the *test set*). In other words, one aims to find a classifier that *generalizes* well and not just achieve optimal error by fitting to idiosyncrasies in the training set – a phenomenon known as *overfitting*. This is especially important when training DNNs in scenarios with mislabeled data; modern DNNs have enough capacity to fit any random permutation of labels (Zhang et al., 2016), and will do just that unless the model is either constrained in some way or trained in a noise-aware manner. Constraining the capacity of a model to prevent overfitting is called *regularization* and done using methods like L_2 norm penalty (Goodfellow et al., 2016) and dropout (Srivastava et al., 2014); these are useful for improving generalization even when one does not have to deal with label noise, since there can be other quirks in the training data. Noise-aware methods are especially designed to make DNN-training robust to label noise by incorporating the presence of noise either into the loss function or the model; most of the literature review in this chapter will de devoted to such methods.

2.3 Learning in the Presence of Label Noise: Problem Setting and Taxonomy

In the presence of label noise, the learner does not always have across entropyss to the true labels. We denote the clean dataset by $\mathcal{D} = \{(x_n, y_n), i = 1, \dots N\}$ drawn from the distribution $p(x, y)$ and the noisy dataset by $\tilde{\mathcal{D}} = \{(x_n, \tilde{y}_n), n = 1, \dots N\}$ drawn from $p(x, \tilde{y})$. In the latter case, we assume some fraction η of the labels have been corrupted though it is not known a priori which of the labels are in error and in the most general case, the noise rate η is also not known. In certain situations, we use η_i to denote the fraction of class i samples that are mislabeled, and additionally, $\eta_{i,j}$ to represent the fraction that have been mislabeled as as class j, with $\sum_{j \neq i} \eta_{i,j} = \eta_i$.

Noise is termed *symmetric* or *uniform* if $\eta_i = \eta$, and $\eta_{i,j} = \frac{\eta_i}{K-1}, \forall j \neq i, \forall i$. This model of label noise is often used in the literature, even though real-world noise is seldom symmetric.

For more realistic modeling, many noise-aware algorithms model *class-conditional* noise using η_{ij}, which allows for the *asymmetric* case. In this setting, it is usually assumed that the dependence of label noise on a sample x is only through the class labels i and j; in other words, dependence of noise on features is not modeled explicitly.

In the work that we review in the following sections, label noise is aways considered to be a stochastic process, and labeling errors are assumed to be independent of each other – this is based on the "random classification noise" (RCN) model introduced in an early work on label noise (Angluin and Laird, 1988). To determine the efficacy of a described method, in most work, label noise is artificially introduced into a clean dataset by randomly flipping the labels of the true class to another class. The resulting label noise depends on the specific noise model, which can be either symmetric or asymmetric; we will make the setting clear when describing the work. However, we will also review recent works where results are reported on real-world noisy datasets.

Loosely following the noise-classification scheme used in (Frénay and Verleysen, 2014), we divide the various approaches to dealing with label noise in deep learning into three broad types. The first type of approach relies on loss functions, or training procedures, that are intrinsically robust to label noise making learning less sensitive to the presence of mislabeled data; we refer to this as *intrinsic noise-robustness*. The second type explicitly models label noise directly either in the loss function, or inside the model; we refer to this as the *noise-aware* approach. A third class of algorithms attempts to identify mislabeled data in a first pass by observing loss function dynamics while training, and then trains on the cleaner set either simultaneously using a second model, or in a downstream step; we refer to this as the *noise filtering* approach (also referred to as *data cleaning* in some works).

A note on taxonomy: the scheme used in (Frénay and Verleysen, 2014) refers to the first type as "noise robust" and the second type as "noise tolerant". However, noise tolerant algorithms make the learning robust to noise, and thus the distinction between the two isn't clear when using such a naming scheme. Further, many papers that model noise – and thus fall into the noise-tolerant category according to (Frénay and Verleysen, 2014) – describe

their methods as being robust to noise, further blurring the distinction between tolerance and robustness; hence, we prefer to use the naming scheme described above.

2.4 Intrinsic Noise Robustness

Early work by (Angluin and Laird, 1988) introduced the RCN model and studied the effect of label noise on binary classification. The authors showed that as long as the noise rate $\eta < 1/2$, a classifier is able to learn the correct concept for classification, and provided an upper bound on the empirical risk of the learned classifier. Investigation of noise-robustness in the deep learning setting is a more recent phenomenon. The robustness of various loss functions for training DNNs was studied in (Ghosh et al., 2017). First, an empirical risk is defined in the label-noise scenario,

$$R_{\mathcal{L}}^{\eta}(f) = \mathbb{E}_{\mathbf{x},\tilde{y}}\left[\mathcal{L}\left(f(\mathbf{x}), \tilde{y}\right)\right] \tag{2.8}$$

which is similar to the $\mathcal{L}-$risk under the noise-free case, but now over the distribution $p(x, \tilde{y})$; the loss function remains unchanged from the corresponding noise-free setting.

Noise robustness as follows: Let f^* be the minimizer of $R_{\mathcal{L}}(f)$ (noise-free setting). A function $\mathcal{L}$ is said to be noise tolerant if the minimizer f_{η}^* of $R_{\mathcal{L}}^{\eta}(f)$ has the same probability of misclassification as that of f^*. The authors examine the robustness of cross entropy, Mean Absolute Error (MAE) and Mean Square Error (MSE) . The authors prove that MAE, defined as

$$\mathcal{L}_{MAE}\left(f(\boldsymbol{x}), \boldsymbol{e}_j\right) = \left\|\boldsymbol{e}_j - f(\boldsymbol{x})\right\|_1 = 2 - 2f_j(\boldsymbol{x}) \tag{2.9}$$

is robust to label noise, and further, the bounded nature of MAE and MSE makes them more robust than an unbounded function like cross entropy. Additionally, the authors also establish an upper bound of $1 - \frac{1}{K}$ on the class-wise noise-rate for noise tolerance to hold in a multi-class scenario[2]

[2]Intuitively this just means that the most likely label for a given class is the true label. If this condition is not met, the DNN – or any other learner that uses ERM – simply learns whatever the most likely class is.

However, empirical results in the paper show that all three cases are significantly susceptible to label noise, even though cross entropy is the hardest-hit. While MAE is theoretically robust, training a DNN using MAE is slow due to weak and saturating gradients, and in a practical setting, hardly ever used, even in the presence of label noise. A better option is to simply train using cross entropy and entrust label-noise robustness to regularization techniques such as dropout (Srivastava et al., 2014) which have been shown to afford a certain amount of robustness to mislabeled data (Arpit et al., 2017).

Noting the above problems with MAE, a modified loss, inspired by the Box-Cox transformation from robust statistics(Sakia, 1992) was proposed in (Zhang and Sabuncu, 2018) as follows:

$$\mathcal{L}_q\left(f(\boldsymbol{x}), \boldsymbol{e}_j\right) = \frac{\left(1 - f_j(\boldsymbol{x})^q\right)}{q} \tag{2.10}$$

where $q \in (0, 1]$. The authors show how this is a generalized version of cross entropy and MAE. L_q is equivalent to cross entropy for $\lim_{q \Rightarrow 0} \mathcal{L}_q$ and becomes MAE when $q = 1$. q is a hyper-parameter that has to be tuned depending on the noise-level by monitoring a separate validation set. Training this way, the authors report significantly improved noise-robustness on a variety of benchmarks.

A significantly different approach towards robustness by enforcing self-consistency on a DNNs predictions is proposed in (Reed et al., 2014). Unlike the earlier described approaches that do not change the training labels, here the training labels are dynamically updated at each iteration by convexly combining the model's current predictions and the given noisy labels. The idea here is that the DNNs own predictions – as it learns on the non-noisy subset – are also taken into account, and in that sense is similar to the self-training and bootstrapping paradigms used in semi-supervised learning[3] (Chapelle et al., 2006). The resulting objective is also consistent with the minimum-entropy models studied in (Grandvalet and Bengio, 2005) that encourages classification boundaries to lie in low-density regions of $\mathcal{X}$.

---

[3]Semi-supervised learning as a strategy for dealing with label noise will be explored in much more detail in Chapter 4

Yet another approach that falls in the self-training category is described in (Tanaka et al., 2018) where an alternating optimization method is employed as follows: train the network as usual, but also alternate it with updating the training labels such that the resulting labels minimize the cross-entropy with the predictions of the current classifier. Similar to (Reed et al., 2014), this is again based on the approach of not relying entirely on the given training labels due to the presence of label noise. Like all self-training procedures, this algorithm is also prone to degenerate solutions where all labels are given the same class; to prevent this additional regularizers are used that use knowledge of prior distribution of classes.

The above do not explicitly model noise rates in either the loss function or modify the DNN in some manner. As mentioned earlier, standard regularization techniques such as dropout (Srivastava et al., 2014) and L^2-regularization (Goodfellow et al., 2016) can also be employed to prevent general overfitting both to label noise and feature noise; we do not discuss those in detail here. Most methods in the label-noise literature attack the label-noise problem by explicitly modeling noise, and we cover such work in the next section.

2.5 Noise-Aware Approaches

In the presence of label noise, minimizing the cross-entropy between predictions and training labels might no longer result in an optimal classifier due to the presence of wrong labels. As mentioned earlier, in such cases, the resulting classifier ends up overfitting the noisy data leading to poor generalization. To deal with this, noise-aware algorithms incorporate the noise model either in the loss function, or as a separate layer in the DNN, with the idea being that modeling the noise this way allows learning on the likely true labels.

A noise-aware robust loss function was proposed in (Natarajan et al., 2013) in a binary class setting with class-conditional noise. The authors propose a *noise corrected* surrogate loss $\hat{\mathcal{L}}$ to the $0-1$ loss that is weighted by the noise rates, such that empirical risk from minimization of $\hat{\mathcal{L}}$ on the noisy data was equivalent to minimization of $\mathcal{L}$ on the clean data i.e, satisfying the noise-robustness condition defined in (Ghosh et al., 2017). The authors assume knowledge of the class conditional noise η_{-1}, η_{+1} and provide bounds on the empirical risk in

the case of convex, shallow learners like support vector machines and logistic regression.

The above line of work was extended in (Liu and Tao, 2015) that showed how a re-weighting method based on noise rates could be applied to *any* surrogate loss function, again in a binary class setting using shallow learners. Additionally, this work also describes a technique for estimating an upper bound on the label noise rates η_{-1}, η_{+1} based on observing the class-conditional predictions $p(\hat{y}|x)$ of the classifier trained on noisy data; this was observed to work well in practice on small binary-class datasets(Asuncion and Newman, 2007).

A re-weighting scheme specifically in the context of training DNNs was proposed in (Ren et al., 2018), where, at every iteration a loss on a *clean validation set* was used to determine how the loss on the training samples should be weighted. Indeed, as we shall see, the idea of using a clean or trusted dataset – if one exists – to guide learning in the presence of noise is one that is employed in many noise-aware algorithms.

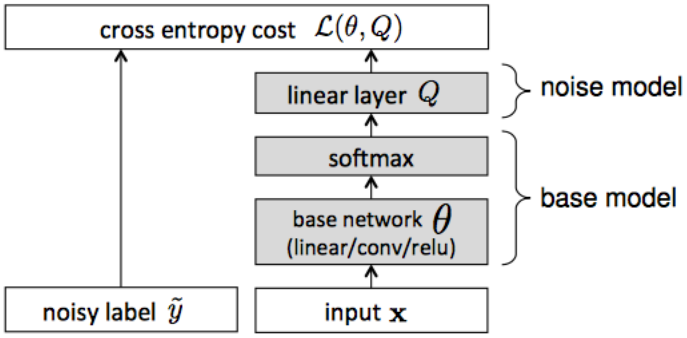


Figure 2.1: A constrained linear layer inserted between softmax layer and loss, with the weight matrix of this layer modeling noise. Output probabilities are transformed from the base model into a distribution that better matches the noisy labels. Figure reproduced from (Sukhbaatar et al., 2014)

The work in (Sukhbaatar et al., 2014) was the first to propose augmenting the DNN with a *noise-aware layer*. The authors propose modeling noise by learning a label flipping matrix Q, where $Q_{i,j}$ is the learned probability of class i incorrectly being flipped to class j, i.e $Q_{i,j} = p\left(\tilde{y} = j|y^* = i\right)$. Q is added to the base DNN model $\mathbf{s}(\Theta)$ after the softmax layer

(Figure 2.1), and is learned during training. The combined model is thus parametrized on Θ and Q, and trained with the following loss

$$\mathcal{L}(\theta, Q) = -\frac{1}{N}\sum_{n=1}^{N} \log \hat{p}\left(\tilde{y} = \tilde{y}_n | \mathbf{x}_n, \theta, Q\right) = -\frac{1}{N}\sum_{n=1}^{N} \log\left(\sum_i Q_{i,\tilde{y}_n} f_i\left(\mathbf{x}_n, \theta\right)\right) \tag{2.11}$$

where $\tilde{y}_n$ is the given noisy label for data sample $\mathbf{x}_n$, $\hat{p}$ is the output of the combined model and $f\left(\mathbf{x}_n, \theta\right)$ is the output of the base model. Note that unless regularized, the base model is perfectly capable of absorbing the noise, resulting in Q degenerating to the identity matrix. To prevent this, and to force Q to converge to the true probability matrix Q^*, a *diffusion process* is enforced by regularizing the norm of $tr(Q)$, which encourages mass to be away from the diagonal terms; a fixed updating schedule and careful choice of weight decay parameters is also needed to learn Q^*.

Noting the difficulty of estimating Q while training, (Patrini et al., 2017) propose a two-phase approach that combines loss correction and noise-layer estimation, but decouples the two. In the first step, an estimate of the noise matrix, $\hat{Q}$ is learned by training the DNN on the noisy data and observing the output probability distribution of a class sample, using the winning class as the ground truth. This is similar to the approach described earlier in (Liu and Tao, 2015), but in a multi-class setting. Then Q is fixed and a loss correction is applied on the model predictions while training using ERM. The authors propose two forms of correction: *backward* loss correction,

$$\ell^{\leftarrow}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = Q^{-1}\ell(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) \tag{2.12}$$

that makes the model match the training data by applying Q^{-1} to the loss calculated on model predictions; and *forward* loss correction that directly changes model predictions by passing the softmax output through the noise layer as done in 2.11. Experiments indicated that the forward method often performed better, but not always. Further, as expected, using $\hat{Q}$ results were significantly worse than when the true noise matrix Q^* was used. Recall that

a DNN can easily overfit training data, and thus training on a noisy set and then observing model predictions on the same set (the authors do not mention using a clean validation set) can lead to significant errors in $\hat{Q}$. However, the method did establish a new state-of-the-art on a real world noisy image dataset (Clothing 1M (Xiao et al., 2015)).

Using a small clean set – "trusted data"– to get a better estimate of Q was precisely what was done in (Hendrycks et al., 2018b). As in (Patrini et al., 2017), a classifier is trained on the potentially noisy set, but the *predicted frequency* of various classes on the *clean set* is used to determine $\hat{Q}$ instead of the softmax scores as done in (Patrini et al., 2017). Both the noisy set and clean set are used in the second stage of training that employs loss correction using $\hat{Q}$. Training this way, the authors report significantly improved robustness even in the presence of very high levels of noise (80%) on various image datasets.

Yet another work that uses a clean data set is described in (Li et al., 2017b). An auxiliary model Π is first trained on a clean dataset; both the predictions from Π and the noisy training labels are then convexly combined and used as the training labels to train a primary model. An optimal setting for the weight λ is derived using an ERM argument. The authors additionally refine the process by constructing the label flipping matrix Q using a "knowledge graph" $\mathcal{G}$ that encodes semantic relationships between classes. $\mathcal{G}$ provides additional information on which classes might be confused with each other and is used to prevent the predictions from the auxiliary model from being too confident. In other words, even though the auxiliary model is trained on a clean dataset, some uncertainty is also added to its predictions. Results in the paper indicate that this provides additional robustness against label noise.

The presence of a human-scrubbed clean dataset is also leveraged in (Veit et al., 2017). A small subset of $\hat{\mathcal{D}}$ is cleaned and then used to train a shallow *label cleaning* network G where, in addition to the image features (computed using a convolutional pre-preprocessing step), the original noisy labels *are also used as input features* for learning; this allows G to learn a mapping between the noisy and clean labels, conditioned on image features. G's predictions on the larger noisy set $\hat{\mathcal{D}}$ are used to train an image classification network. The setup is

rather elaborate involving multiple modules and is framed as a multitasking learning process, with the whole apparatus being jointly trained.

Continuing with noise-layer methods, another such approach is described in (Goldberger and Ben-Reuven, 2016). Treating the true labels as latent variables, first an Expectation-Maximization (EM) (Dempster et al., 1977) approach is discussed, alternatively computing the true labels E-step) and the model parameters Q and Θ (M-step). To avoid the full cost of EM– where a DNN has to be trained at every M-step – a joint-training approach is proposed where Q and Θ are simultaneously learned by adding a *second softmax layer* to the network. In this regard, the work is very similar to the previously discussed work with the only difference being that the noise adaptation layer is passed through a softmax function instead of being a purely linear layer. Also, Q is initialized using the predictions of the network first trained on $\tilde{D}$ without the noise adaptation layer. The authors also address a computational issue inherent to methods that use Q – a full Q matrix is $O(K^2)$ in memory requirements, and is not scalable to datasets involving a large number of classes. To address this, the connections between the final layers were pruned – only the top m classes that were actually confused with each other in the first step have connections between the two softmax layers, which greatly sparsifies Q.

An EM approach was also proposed in (Xiao et al., 2015), where two types of latent variables were added: one representing the true labels, and another representing the category of noise type – no noise, symmetric noise and class conditional noise. It was observed that a transfer learning approach – where a pre-trained network on a different image domain was then fine-tuned on a clean dataset in the relevant domain – performed worse than training a network with large, although noisy data directly on the domain of interest. An important contribution of this paper was the introduction of a large, real-world noisy dataset consisting of one million clothing images. The images were first scraped from online sources using weak tags, and the authors manually cleaned the test and validation sets, and in the process, were able to estimate the noise on the training set to be approximately 40%; this set has subsequently been used in many label-noise works.

2.6 Noise Filtering

The approaches described in the previous section essentially attempt to learn on the correct labels (as estimated by the DNN during training) by incorporating a model of label noise, or using noise-robust training approaches, but still essentially train on the entire dataset. It is inevitable that while doing so, many correctly labeled instances will also be wrongly corrected. To avoid this "false correction" phenomena, an alternative approach is to actively identify samples that are likely to have been mislabeled and mitigate their effect on learning by suppressing their gradients, or dropping them from the training set altogether[4]. The problem then becomes one of reliably identifying these mislabeled samples, and *filtering* these out during learning. Early work (Brodley and Friedl, 1999) showed that this approach can work well in classical machine learning using an ensemble of learners. In deep learning, too, this has shown to work by exploiting the dynamics of learning in DNNs, which we discuss below.

It has been observed that while training modern DNNs – which are models with sufficiently high representational capacity to overfit training data – overfitting happens during the *later* stages of training (Goodfellow et al., 2014b), a phenomena that was observed even in the early days of neural network research (Geman et al., 1992). Both training error and validation error first decrease steadily, but over time train error continues to decrease while validation error begins to rise again (Figure 2.2) implying that additional training is simply fitting to quirks in the training data.

To prevent overfitting, a frequently employed method is "early stopping": training is stopped when validation error is seen to rise[5] and the model with the lowest validation error is used for subsequent testing and deployment. Early stopping is essentially a form of regularization since it constrains the number of training steps the model can take before fitting the data. Because of its simplicity and effectiveness, early stopping has been one of the

[4]Recent methods adopt a semi-supervised learning approach where only the *labels* of the potentially noisy samples are dropped; we cover such methods in Chapter 4

[5]Because of the stochastic nature of training, one usually has to observe this behavior over a number of iterations before deciding to stop

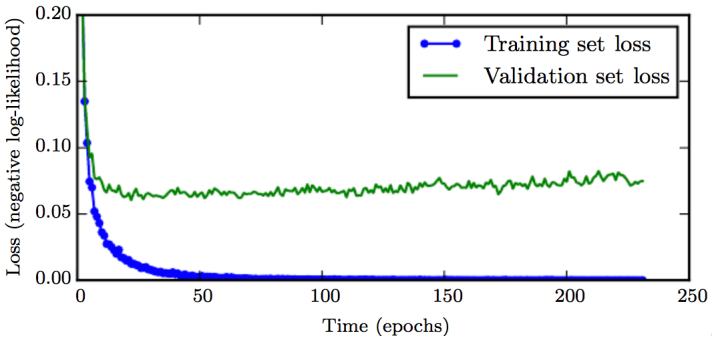


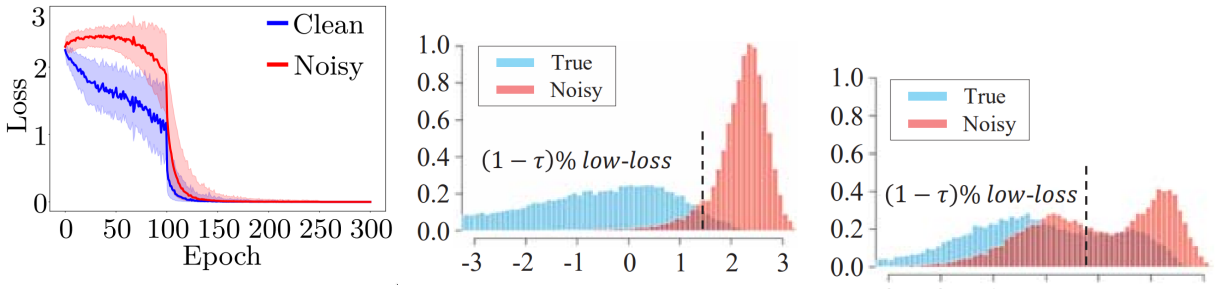
Figure 2.2: Training error decreases consistently but the validatin error eventually increases, suggesting that DNN is overfitting to the traiing data. Figure reproduced from (Goodfellow et al., 2016)

most widely used forms of regularization, both in modern deep learning (Goodfellow et al., 2014b) and with older feed-forward networks (Prechelt, 1998).

The effectiveness of early stopping suggests that DNNs are *learning patterns in the data before overfitting to noise*, because fitting to consistent patterns in the data might, in some sense, be easier than fitting to the more arbitrary patterns of noise. Indeed, in (Zhang et al., 2016) it was observed that training took longer to converge in the presence of severe label noise, and experiments in (Arpit et al., 2017) indicated that DNNs appeared to be learning patterns before *memorizing*[6] the noise.

Indeed, when training with corrupted data it has been observed that, in the early stages of training, the train loss on corrupted samples is often much higher than the loss on correctly labeled data, before converging in the overfitting phase (Figure 2.3) At certain points in training, in the symmetric loss scenario, the losses even appear to come from *two distinct loss populations* (Figure 2.3b) suggesting an easy way to filter out noisy samples. However, real-world noise is seldom symmetric, and in the more realistic pair-noise scenario (where only pairs of similar classes are confused with each other), the separation is much less

[6]There is no accepted formal definition in the literature as to what it means to *memorize*, even though this terms is frequently employed when discussing the capacity of DNNs. (Arpit et al., 2017) give an *operational definition* of memorization as the *"behavior exhibited by DNNs trained on noise"*.



(a) 80% Symmetric Noise, CIFAR-10 (b) 40% Symmetric Noise, CIFAR-100 (c) 40% Pair Noise, CIFAR-100

Figure 2.3: DNNs learn differently on clean and mislabeled data. (a) Train loss evolution on clean and noisy samples on CIFAR-10 under 80% label noise. Reproduced from (Arazo et al., 2019). (b) Loss histograms on clean and noisy data for 40% label noise on CIFAR-100 at train accuracy of 50% for symmetric noise and, (c) for pair noise. Reproduced from (Song et al., 2019).

pronounced(Figure 2.3c) . Nevertheless, the train loss seems to be somewhat indicative of data corruption, and a slew of recent methods attempt to tackle label noise by exploiting this phenomena; we discuss these below.

Noting that the updates during the later stages of learning are likely due to label noise, (Malach and Shalev-Shwartz, 2017) propose a pair-learning approach — two networks are trained simultaneously, learning for a fixed number of iterations as usual, but in the later stages only updating if there is disagreement between the two. The gradients are only back-propagated on these samples, and not just when there is disagreement between output and train label, i.e disagreement between the peer networks is trusted more than a disagreement with the training labels. Analysis is provide on perceptrons for linearly separable data and empirically this works for upto 40% label noise for DNNs. The number of epochs to train before mistrusting updates from the given data is a hyerparameter and depends on the amount of label noise present.

Another pair-training approach called "co-teaching" was proposed in (Han et al., 2018) that also exploits the fact that low-loss samples are likely to be cleaner, and the networks

give more weight to these samples. However, to prevent reinforcement of biases, the networks *teach each other*: at each iteration, the optimization on each network's parameters are with respect to the loss on the samples that the other network considers clean (low loss); in other words, self-reinforcement of biases is mitigated by a form of "peer review". The method crucially depends on what is considered low-loss: initially the criterion is generous to allow learning on most samples, but towards the later stages, to prevent overfitting, the gradient updates are done on fewer and fewer samples. However networks, when trained long enough, eventually converge and collapse to self-training mode. An improved variant of co-teaching, called "co-teaching+" was proposed in (Yu et al., 2019) that combines co-teaching with the disagreement approach of (Malach and Shalev-Shwartz, 2017) which was shown to prevent the collapse into self-training mode.

A teacher-student model of pair-training, called "MentorNet" is explored in (Jiang et al., 2018b) where the main idea is inspired by curriculum learning (CL) (Bengio et al., 2009): learning is ordered by the difficulty of the samples, with easy samples being learnt before the harder ones. In traditional CL, the curriculum is pre-determined, usually by a human. In the MentorNet framework, this is determined by the teacher network by observing the training loss, and based on this, data is then re-weighted to train the student network. This prioritizes learning on easier samples by the student network, and presumably suppresses learning on the noisy – and thus harder – samples.

The separability of losses on the noisy and corrupted data was exploited to train a beta-mixture-model (BMM) in (Arazo et al., 2019) to identify the two populations. The loss on the samples that are estimated to be from the noisy population are down-weighted during training, while the BMM parameters are estimated using an EM approach. Further significant improvements are seen when the training data is augmented using techniques like mixup (Zhang et al., 2017) that has been shown to have an inhibitive effect on overfitting. State-of-art results were reported on various image benchmarks using this approach.

A low-loss approach based on alternatively selecting samples with low loss and then training the model on the remaining samples was proposed in (Shen and Sanghavi, 2019).

The method – called "iterative trimmed loss" – is inspired from a robust statistical concept of *trimmed loss* that attempts to fit a model by selectively minimizing the loss on a fraction of the training samples. Selecting the samples is a combinatorially hard problem and generally intractable; the authors attempt to approximate this via the above iterative approach using low-loss as the selection criterion.

The lack of clear separability between noisy and clean samples in real-world noisy settings was noted in (Song et al., 2019), which is basically an outcome of the fact that many cleanly labeled samples are hard-to-learn samples, and thus might have high loss during training. Indeed, recent work (Zhou and Bilmes, 2018) has exploited the "hardness" of samples to improve convergence of training; however the presence of label noise wasn't considered in that work. The work in (Song et al., 2019) considers this phenomena in the presence of label-noise, and to improve noise-filtering, a "selective refurbishing" of noisy samples was proposed. The key idea here is that samples that are predicted wrongly, but *consistently*, are considered refurbishable and they are then assigned the most frequently predicted label (label correction). This is then brought back into the fold of data on which learning takes place, resulting in improved performance.

Recent theoretical analysis has begun to shed light on the above "small-loss" approaches, and the related phenomena of early stopping being an effective regularizer against label noise. In a paper titled "Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks" (Li et al., 2019), the authors show that in order to overfit to noise, the network must stray far from the initialization point. If there are learnable patterns in the data, these are learned early on in training. The authors tie this to the singular values of the Jacobian of the DNN: large singular values of the Jacobian correspond to the gradients on the clean data, and SGD learns along these directions first. The smaller singular vaiues, on the other hand, correspond to the mislabeled data; fitting to these happens later during training.

The fact that early stopping learns a parameter vector not too far away from the initialization – which is usually close to the origin – is consistent with the effect of regularization

methods like L_2 penalty that prevent overfitting by constraining the norm of the parameters; this connection between early stopping and L_2 regularization was explicitly made in another recent work (Hu et al., 2019). While these theoretical works provide some justification to why early stopping is effective against label noise, the analyses has been mostly limited to shallow networks, due to the complexity of modern DNNs. Nevertheless, the "small-loss" approach works well in practice, and the abstention framework that we introduce in this next chapter also exploits this, but in a different way.

2.7 Conclusion

Deep learning in the presence of label noise is a relatively new, although very active, area of research. In this chapter we attempted to summarize the major paradigms for tackling label noise, although a comprehensive summary is difficult due to the fast and ever growing body of literature. The state-of-the-art methods, as of this writing, are either those using various versions of the noise filtering approach or using a trusted data set; simply using standard, although robust, loss functions does not match the performance of the other classes of methods. Most methods report experimental results on synthetically induced random noise on existing clean datasets. This is only a rough approximation of real-world label noise which is usually dependent on features present in the image. And thus it is remarkable that, among the current crop of methods, robustness to feature-dependent label noise is not modeled; we introduce precisely such a framework in the next chapter.

Chapter 3

COMBATING LABEL NOISE USING ABSTENTION

The art of knowing is knowing what to ignore.

— Rumi

We introduce the "Deep Abstaining Classifier", a deep neural network that learns robustly in the presence of noisy labels using a novel, abstention-based approach. State-of-the-art results are shown for multiple types of label noise on various image benchmarks. This was joint work with Tanmoy Bhattacharya, Jeff Bilmes, Gopinath Chennupati and Jamaludin Mohd-Yusof; the chapter is based on our paper that appeared in the proceedings of the International Conference on Machine Learning, 2019 (Thulasidasan et al., 2019a).

3.1 Introduction: The Deep Abstaining Classifier

In this chapter we introduce a novel *abstention* -based mechanism to combat label noise while training deep neural networks in the presence of label noise. This allows the DNN to abstain – or defer learning – on confusing or noisy samples while continuing to learn and improve classification performance on the non-abstained samples. Unlike the approaches covered in the previous chapter, we show that our loss function also *enables learning representations* for features that might correlate with unreliable labels.

Most of the theoretical and empirical investigations into abstention (or rejection) classification systems have been studied in a post-processing setting – i.e., a classifier is first trained as usual, and an abstention threshold is determined based on post-training performance on a calibration set; the DNN then abstains on uncertain predictions during inference. Abstention classifiers during prediction have been proposed for shallow learners (Chow, 1970; Cortes et al., 2016; Fumera and Roli, 2002) and for multilayer perceptrons (De Stefano et al., 2000)

and also recently in the context of deep networks (Geifman and El-Yaniv, 2017) but in contrast to these, the method described in this chapter employs abstention during *training* as well as inference. This gives the DNN an option to abstain on a confusing training sample thereby mitigating the misclassification loss but incurring an abstention penalty. We provide analytical results on the loss function behavior that enable dynamically setting abstention rates based on learning progress during training. Our formulation ensures that the DNN continues to learn the true class even while abstaining, progressively reducing its abstention rate as it learns on the true classes, finally abstaining on only the most confusing samples.

We demonstrate the advantages of such a formulation under two different label-noise scenarios: first, when labeling errors are correlated with some underlying feature of the data (systematic or structured label noise), abstention training allows the DNN to learn features that are indicative of unreliable training signals and which are thus likely to lead to uncertain predictions. This kind of representation learning for abstention is useful both for effectively eliminating structured noise and also for interpreting the reasons for abstention.

Second, we show how an abstention-based approach can be used as a very effective *data cleaner* when training data contains arbitrary (or unstructured) label noise. A DNN trained with an abstention option can be used to identify and filter out noisy training data leading to significant performance benefits for downstream training using a cleaner set. To summarize, the contributions described in this chapter are:

- The introduction of abstention-based training as an effective approach to combat label noise. We show that such a deep abstaining classifier (DAC) enables robust learning in the presence of label noise.

- The demonstration of the ability of the DAC to learn features associated with systematic label noise. Through numerous experiments, we show how the DAC is able to pick up (and then abstain on) such features with high precision.

- Demonstration of the utility of the DAC as a highly effective *data cleaner* in the

presence of arbitrary label noise. We provide results on learning with noisy labels on multiple image benchmarks (CIFAR-10, CIFAR-100 and Fashion-MNIST) that are competitive and in many cases, significantly improve performance compared to existing methods.

Code for the DAC and experiments in this chapter are available at https://github.com/thulas/dac-label-noise

3.2 Loss Function for the DAC

We assume we are interested in training a k-class multi-class classifier with a deep neural network (DNN) where x is the input and y is the output. For a given x, we define $p_i = p_w(y = i|x)$ (the probability of the ith class given x) as the i^{th} output of the DNN that implements the probability model $p_w(y = i|x)$ (using a softmax function as its final layer) where w is the set of weight matrices of the DNN. For notational brevity, we use p_i in place of $p_w(y = i|x)$ when the input context x is clear.

The standard cross-entropy training loss for DNNs then takes the form $\mathcal{L}_{\text{standard}} = -\sum_{i=1}^{k} t_i \log p_i$ where $t_i \in 0, 1$ is the target for the current sample. The DAC has an additional $k + 1^{\text{st}}$ output p_{k+1} which is meant to indicate the probability of abstention. We train the DAC with the following modified version of the k-class cross-entropy per-sample loss:

$$\mathcal{L}(x) = (1 - p_{k+1}) \left(-\sum_{i=1}^{k} t_i \log \frac{p_i}{1 - p_{k+1}} \right) + \alpha \log \frac{1}{1 - p_{k+1}} \tag{3.1}$$

The first term is a modified cross-entropy loss over the k non-abstaining classes. Absence of the abstaining output (i.e., $p_{k+1} = 0$) recovers exactly the usual cross-entropy.The second term penalizes abstention and is weighted by $\alpha \geq 0$, a hyper-parameter expressing the degree of penalty. If α is very large, there is a high penalty for abstention thus driving p_{k+1} to zero and recovering the standard unmodified cross-entropy loss; in such case, the model learns to never abstain. With α very small, the classifier may abstain on everything with impunity since the adjusted cross-entropy loss becomes zero (since $\lim_{p \to 0} p \log p = 0$) and it

does not matter what the classifier does on the k class probabilities. When α is between these extremes, things become more interesting. Depending on α, if it becomes hard during the training process for the model to achieve low cross-entropy on a sample, then the process can decide to allow the model to abstain on that sample.

Even though the model can opt to abstain on a difficult sample (measured by the cross-entropy loss), the formulation of the loss function in Equation 3.1 allows learning on the given class, *even in the presence of abstention.* This is by design, since the given label might indeed be the correct label and it is generally not possible to know whether a high cross-entropy loss is due to the inherent learning difficulty of a correctly labeled sample, or due to label noise; we discuss how this distinction can be made in Section 3.4 by tracking the model's performance on a clean validation set. In any case, Equation 3.1 allows the model to learn on the given label by ensuring that the pre-softmax activation of a given class continues to increase while training.

Let j, $(1 \leq j \leq k)$ be the given class for x. During gradient descent, learning on the true class takes place if $\frac{\partial \mathcal{L}}{\partial a_j} < 0$, where a_j is the pre-activation into the softmax unit of class j. This ensures that a_j continues to increase during training; in the following theorem, we prove that this is indeed the case.

Theorem 1. *For the loss function $\mathcal{L}$ given in Equation 3.1, if j is the given class for sample x, then as long as $\alpha \geq 0$, $\frac{\partial \mathcal{L}}{\partial a_j} \leq 0$ (where a_j is the pre-activation into the softmax unit of class j).*

Proof. Consider again the loss function defined in Equation 3.1 for a sample x.

$$\mathcal{L}(x) = (1 - p_{k+1}) \left(- \sum_{i=1}^{k} t_i \log \frac{p_i}{1 - p_{k+1}} \right) + \alpha \log \frac{1}{1 - p_{k+1}}. \tag{3.2}$$

We assume that we are training with hard labels, in which case $t_j = 1$ and $t_i = 0 \; \forall i \neq j$. A straight-forward gradient calculation shows that

$$\frac{\partial \mathcal{L}}{\partial a_j} = -(1 - p_j - p_{k+1}) + p_{k+1} p_j \log\left(\frac{1 - p_{k+1}}{p_j}\right) - \alpha \frac{p_{k+1} p_j}{1 - p_{k+1}} \tag{3.3}$$

Since $\alpha \geq 0$ as per as our assumption, the last quantity in the above expression, $-\alpha \frac{p_{k+1} p_j}{1 - p_{k+1}} \leq 0$

Also note that $(1 - p_j - p_{k+1})$ in the above expression is just the total probability mass in the remaining real (i.e non abstention) classes; denote this by q.

Then we have

$$\begin{aligned}
-(1 - p_j - p_{k+1}) + p_{k+1} p_j \log\left(\frac{1 - p_{k+1}}{p_j}\right) &= -q + p_{k+1} p_j \log\left(\frac{1 - p_{k+1}}{p_j}\right) && (3.4)\\
&= -q + p_{k+1} p_j \log\left(\frac{q + p_j}{p_j}\right) && (3.5)\\
&= -q + p_{k+1} p_j \log\left(1 + \frac{q}{p_j}\right) && (3.6)\\
&\leq -q + p_{k+1} p_j \frac{q}{p_j} && (3.7)\\
&= -q + p_{k+1} q && (3.8)\\
&\leq 0 && (3.9)
\end{aligned}$$

where, in 3.7, we have made use of the fact that $\log(1 + x) \leq x$ for all $x > -1$. Thus $\frac{\partial \mathcal{L}}{\partial a_j} \leq 0$ as desired. □

The above result ensures that, during gradient descent, learning on the true classes persists even in the presence of abstention, even though the true class might not end up be the winning class.

Next, we look at the Jacobian of the loss function with respect to the pre-activation into the other non-abstention classes. For a given sample, let a_i be the pre-softmax activation for

class $i, i \neq j, i \neq K+1$. Then, we can show that

$$\frac{\partial \mathcal{L}}{\partial a_i} = p_i + p_{k+1} p_i \log\left(\frac{1-p_{k+1}}{p_j}\right) - \alpha \frac{p_{k+1} p_i}{1-p_{k+1}} \tag{3.10}$$

For very small α, the above gradient will be positive, meaning activation a_i will *decrease* during training. This is generally desirable, since it is the activation into the given class – i.e., a_j – that should increase during training. However, it is possible that this gradient can be negative under certain combinations of α, p_i, p_j and p_{k+1}. Further it is even possible that the above gradient might be *even more negative* than that of the true class, i.e., under certain conditions $\frac{\partial \mathcal{L}}{\partial a_i} < 0$ *and* $\frac{\partial \mathcal{L}}{\partial a_i} < \frac{\partial \mathcal{L}}{\partial a_j}$. For example, if $p_i = 0.01, p_j = 0.001, p_{K+1} = 0.9$ and $\alpha = 2$, then $\frac{\partial \mathcal{L}}{\partial a_i} < \frac{\partial \mathcal{L}}{\partial a_j}$. In our experimental analysis, we find this to be rarely the case, however and in most cases $\frac{\partial \mathcal{L}}{\partial a_j} < \frac{\partial \mathcal{L}}{\partial a_i}$, which ensures that among the non-abstention classes, it is the mass into the given class – i.e p_j – that increases at the highest rate.

3.2.1 Auto-tuning Abstention Behavior

Let $g = -\sum_{i=1}^{k} t_i \log p_i$ be the standard cross-entropy loss and a_{k+1} be the pre-activation into the softmax unit for the abstaining class. Then we can show that:

$$\frac{\partial \mathcal{L}}{\partial a_{k+1}} = p_{k+1}\left[(1-p_{k+1})\left[\log\frac{1}{1-p_{k+1}} - g\right] + \alpha\right]. \tag{3.11}$$

During gradient descent, abstention pre-activation is increased if $\frac{\partial \mathcal{L}}{\partial a_{k+1}} < 0$. The threshold on α for this is $\alpha < (1-p_{k+1})\left(-\log\frac{p_j}{1-p_{k+1}}\right)$ where j is the true class for sample x. If only a small fraction of the mass over the actual classes is in the true class j, then the DAC has not learned to correctly classify that particular sample from class j, and will push mass into abstention class provided α satisfies the above inequality. This constraint allows us to perform auto-tuning on α during training (Algorithm 1). $\tilde{\beta}$ is a smoothed moving average of the α threshold (initialized to 0), and updated at every mini-batch iteration. We perform abstention-free training (i.e., training with regular cross-entropy) for a few initial epochs (warm-up period L) to accelerate learning, triggering abstention from epoch $L+1$ onwards.

Algorithm 1: α auto-tuning

Input: total iter. (T), current iter. (t), total epochs (E), abstention-free epochs (L), current epoch (e), α init factor (ρ), final α (α_{final}), mini-batch cross-entropy over true classes ($\mathcal{H}_c(P^M_{1\ldots K})$

$\alpha_{set} = False$

for $t := 0$ to T **do**

  if $e < L$ **then**

    $\beta = (1 - P^M_{k+1})\mathcal{H}_c(P^M_{1\ldots K})$

    if $t = 0$ **then**

      $\tilde{\beta} = \beta\{$ // initialize moving average$\}$

    end if

    $\tilde{\beta} \leftarrow (1 - \mu)\tilde{\beta} + \mu\beta$

  end if

  if $e = L$ and **not** α_{set} **then**

    $\alpha := \tilde{\beta}/\rho\{$ // initialize α at start of epoch $L\}$

    $\delta_\alpha := \frac{\alpha_{final} - \alpha}{E - L}$

    $update_{epoch} = L$

    $\alpha_{set} = True$

  end if

  if $e > update_{epoch}$ **then**

    $\alpha \leftarrow \alpha + \delta_\alpha\{$ //then update α once every epoch$\}$

    $update_{epoch} = e$

  end if

end for

At the start of abstention, α is initialized to a much smaller value than the threshold $\tilde{\beta}$ to encourage abstention on all but the easiest of examples learned so far.

Initially, α was set to a constant value throughout training. As long as α meets the abstention threshold, we can ensure that the DAC will learn to push mass into the abstention class. However, the trouble with this approach is that a fixed α will eventually fail to meet this threshold due to loss function dynamics, at which point the DAC stops abstain; we formalize this in the next theorem:

Theorem 2. *For the loss function $\mathcal{L}$ given in Equation 3.1, for a fixed α, and trained over t epochs, as $t \to \infty$, the abstention rate $\gamma \to 0$ or $\gamma \to 1$.*

Proof. If α is close to 0, p_{k+1} quickly saturates to unity, causing the DAC to abstain on all samples, driving both loss and the gradients to 0 (since if $\alpha \approx 0$, and $p_{k+1} \to 1$, then $\frac{\partial \mathcal{L}}{\partial a_j} \to 0$,

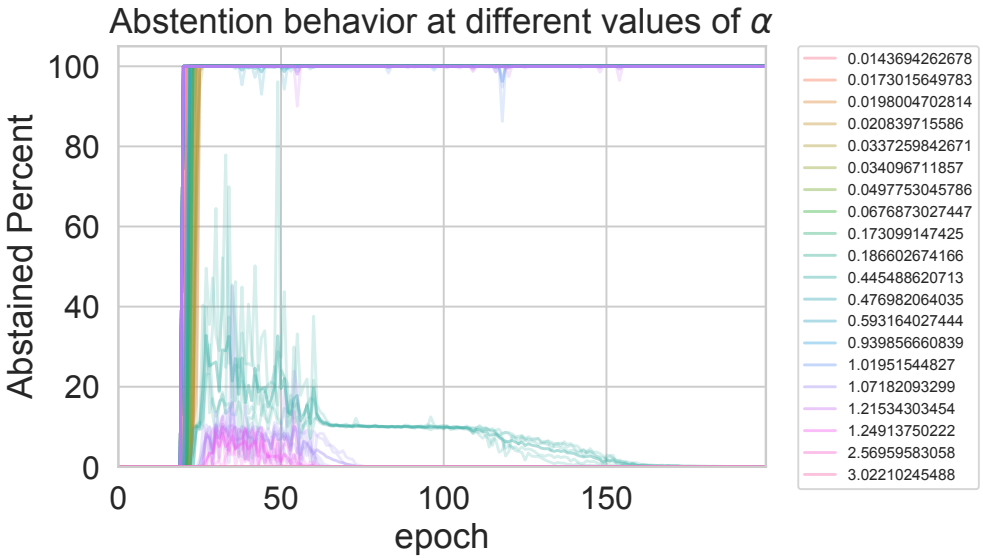


Figure 3.1: DAC behavior (where 10% of the data has corrupted labels) for different fixed α's (indicated by color) illustrating the tendency for all-or-nothing abstention if α is kept constant.

from Equations 3.3 and 3.9), preventing any further learning on the given class. Barring this situation, and given that the gradient $\frac{\partial \mathcal{L}}{\partial a_j} \leq 0$, where j is the given class (see Theorem 1), the condition for abstention in Section 3.2.1 eventually fails to be satisfied. This can be seen as follows: let $\tau = (1 - p_{k+1}) \left(-\log \frac{p_j}{1-p_{k+1}} \right)$ represent the threshold on α for abstention pre-activation (a_{K+1}) to increase. A gradient calculation shows that

$$\frac{\partial \tau}{\partial a_j} = -(1 - p_j - p_{k+1}) + p_{k+1} p_j \log \left(\frac{1 - p_{k+1}}{p_j} \right) \tag{3.12}$$

$$\leq 0 \tag{3.13}$$

where the last result follows from our derivations in Equations 3.4 through 3.7. Here a_j is the pre-activation into the given class, which, from Theorem 1, we know only increases during training. Thus τ continuously decreases during training, and at some point we will have $\alpha > \tau$. After this happens, α no longer satisfies the condition for abstention and probability mass

is removed from the abstention class $K+1$ for all subsequent training, eventually driving abstention to zero. □

Experiments where α was fixed confirm this. Figure 3.1 shows abstention behavior for different values of fixed alpha in an experiment where 10 % of the labels were corrupted; depending on the value of α (which was constant in these experiments), abstention rate asymptotes to 0 or 1. As an alternative strategy to a fixed α, we employ a linear ramping of α that increasingly penalizes abstention on only the most diffcult samples. As the learning progresses on the true classes, abstention is reduced. We will describe a control theoretic approach for stabilizing abstention at a given setpoint in the next chapter, but for all the experiments in this chapter we employ the linear ramping schedule given in Algorithm 1. In the experiments in the subsequent sections, we illustrate how the DAC, when trained with this loss function, learns representations for abstention.

3.3 The DAC as a Learner of Structured Noise

3.3.1 Background

While noisy training labels are usually an unavoidable occurrence in real-world data, such noise can often exhibit a pattern attributable to training data being corrupted in some non-arbitrary or systematic manner. This kind of label noise can occur when some classes are more likely to be mislabeled than others, either because of confusing features or a lack of sufficient level of expertise or unreliability of the annotator. For example, the occurrence of non-iid, systematic label noise in brain-computer interface applications – where noisy data is correlated with the state of the participant – has been documented in (Porbadnigk et al., 2014; Görnitz et al., 2014). In image data collected for training (that might have been automatically pre-tagged by a recognition system), a subset of the images might be of degraded quality, causing such labels to be unreliable[1]. Further, systematic noise can also occur if all the data were labeled using the same mechanism (see discussion in (Brodley and

[1]We assume, in this case, one only has access to the labels, and not the confidence scores

Friedl, 1999)); for a comprehensive survey of label noise see (Frénay and Verleysen, 2014).

In these scenarios, there are usually consistent indications in the input x that tend to be correlated with noise in the labels, but such correlations are rarely initially obvious. Given the large amount of training data required, the process of curating the data down to a clean, reliable set might be prohibitively expensive. In situations involving sensitive data (patient records, for example) crowd-sourcing label annotations might not even be an option. However, given that DNNs can learn rich, hierarchical representations, one of the questions we explore in this paper is whether we can exploit the representational power of DNNs to *learn* such feature mappings that are indicative of unreliable or confusing samples. The loss function formulation encourages the DNN to put mass in the abstention class when the cross-entropy on the true classes is high. Thus features that are consistently picked up by the DAC during abstention should thus have high feature weights with respect to the abstention class, suggesting that the DAC might learn to make such associations. In the following sections, we describe a series of experiments on image data that demonstrate precisely this behavior – using abstention training, the DAC learns features that are associated with difficult or confusing samples and reliably learns to abstain based on these features.

3.3.2 Experimental Setup

For the experiments in this section, we use a deep convolutional network employing the VGG-16 (Simonyan and Zisserman, 2014) architecture, implemented in the PyTorch (Paszke et al., 2017) framework. We train the network for 200 epochs using SGD accelerated with Nesterov momentum and employ a weight decay of .0005, initial learning rate of 0.1 and learning rate annealing using an annealing factor of 0.5 at epoch $60, 120$ and 160. We performing abstention free training during the first 20 epochs which allows for faster training[2] In this section, we use the labeled version of the STL-10 dataset (Coates et al., 2011), comprising of 5000 and 8000 96x96 RGB images in the train and test set respectively, augmented with

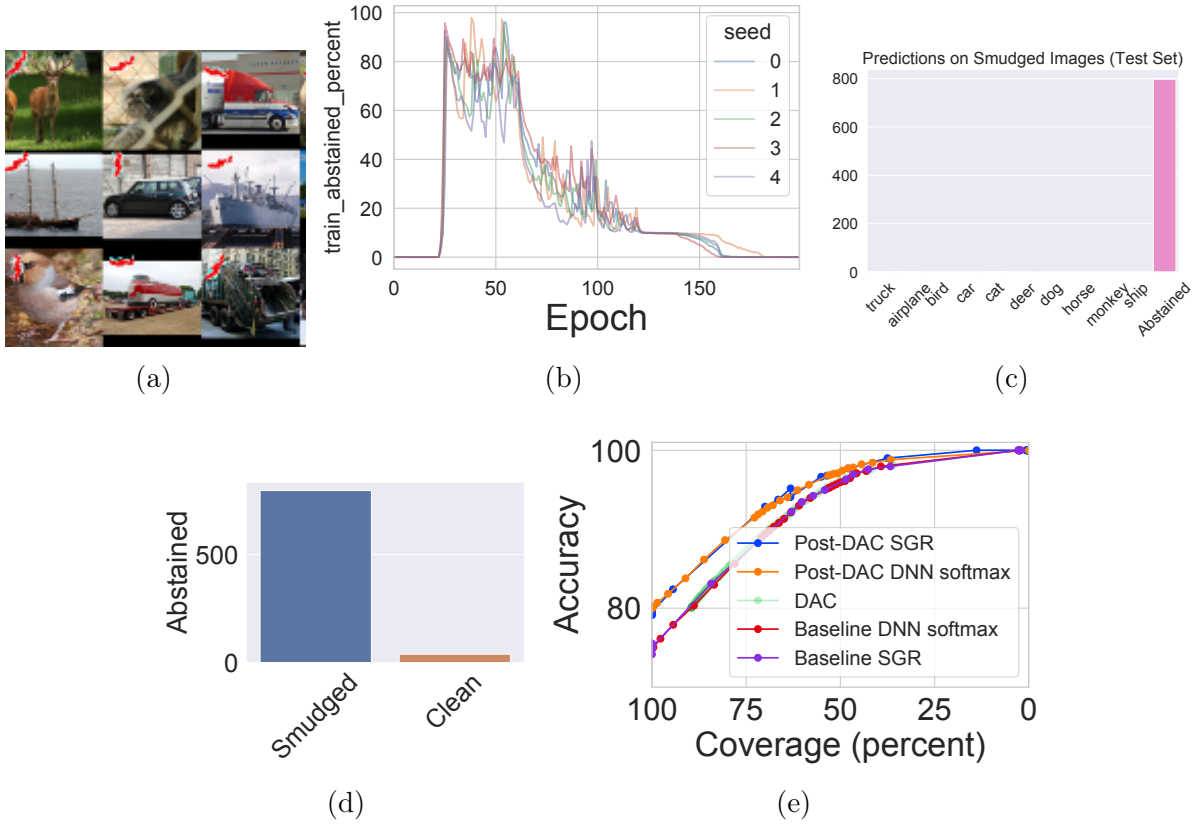
[2]Training with abstention from the start just means we have to train for a longer number of epochs to reach a given abstention-vs-accuracy point.

random crops and horizontal flips during training. We use this architecture and dataset combination to keep training times reasonable, but over a relatively challenging dataset with complex features. For the α auto-update algorithm we set ρ (α initialization factor) to 64 and μ to 0.05. Unless otherwise mentioned, the labels are only corrupted on the training set; depending on the experiment, the features in the validation set might also be perturbed – these will be explicitly noted.

3.3.3 Results: Noisy Labels Co-Occurring with an Underlying Cross-Class Feature

In this experiment we simulate the situation where an underlying, generally unknown feature occurring in a subset of the data often co-occurs with inconsistent mapping between features and ground truth. In a real-world setting, when encountering data containing such a feature, it is desired that the DAC will abstain on predicting on such samples and hand over the classification to an upstream (possibly human) expert. To simulate this, we randomize the labels (over the original K classes) on 10% of the images in the training set, but add a distinguishing extraneous feature to these images. In the experiments in this section, this feature is a *smudge* (Figure 3.2a) that represents the afore-mentioned feature co-occurring with label noise. We then train both a DAC as well as a regular DNN with the usual K class cross-entropy loss. Performance is tested on a set where 10% of the images are also smudged. Since it is hoped that the DAC learns representations for the structured noise occurring in the dataset, and assigns the abstention class for such training points, we also report the performance of a DNN that has been trained on a set where the abstained samples were eliminated (*post-DAC*) at the best-performing epoch of the DAC[3]. Performance is reported in terms of accuracy-vs-abstained (i.e., risk-vs-coverage) curves for the DAC, and the standard softmax threshold-based abstention for the baseline DNNs and post-DAC DNNs. As an additional baseline, we also compare the performance of the recently proposed selective guaranteed risk method in (Geifman and El-Yaniv, 2017) for both the baseline and post-DAC

[3]we describe in Section 3.4 how noisy data is eliminated



(a) (b) (c)

(d) (e)

Figure 3.2: (a) A sample of smudged images on which labels were randomized.(b) Abstention percentage on training set as training progresses (c)The DAC abstains on most of the smudged images on the test set (abstention recall) (d) Almost all of the abstained images were those that were smudged (abstention precision) (e) Risk-coverage curves for baseline DNNs, DAC and post-DAC DNN for both softmax-based thresholds and SGR. First-pass training with the DAC improves performance for both softmax and SGR methods.

DNNs that maximizes coverage subject to a user specified risk bound (we use their default confidence parameter, δ, of 0.001 and report coverage for a series of risk values).

Results When trained over a non-corrupted set, the baseline (i.e., non-abstaining) DNN had a test accuracy over 82%, but this drops to under 75% (at 100% coverage) when trained on the label-randomized smudged set (Figure 3.2e). For the DAC, Figure 3.2b shows the abstention progress on the train set. When abstention is initiated after 20 epochs, the DAC

chooses to abstain on all but the easiest samples learned so far, but progressively abstains on less data till the abstention reaches steady behavior between epochs 120 and 150, abstaining on about 10% of the data, representing the smudged images. Further annealing of learning rate (at epoch 160) causes the DAC to go into memorization mode.[4]. However, at the best performing validation epoch, the DAC abstains – with both high precision and recall – on precisely those set of images that have been smudged (Figures 3.2c and 3.2d)! In other words, it appears the DAC has learned a clear association between the smudge and unreliable training data, and opts to abstain whenever this feature is encountered in an image sample. Essentially, the smudge has become a separate class all unto itself, with the DAC assigning it the abstention class label. The abstention-accuracy curve for the DAC (red curve in Figure 3.2e, calculated using softmax thresholds at the best validation epoch, closely tracks the baseline DNN's softmax thresholded curve – this is not surprising, since on those images that are not abstained on, the DAC and the DNN learn in similar fashion due to the way the loss function is constructed. We do however see a strong performance boost by eliminating the data abstained on by the DAC and then re-training a DNN – this post-DAC DNN (blue curve) has significantly higher accuracy than the baseline DNN (Figure 3.2e), and also has consistently better risk-coverage curves. Not surprisingly this performance boost is also imparted to the SGR method since any improvement in the base accuracy of the classifier will be reflected in better risk-coverage curves. In this sense, the DAC is complementary to an uncertainty quantification method like SGR or standard softmax thresholding – first training with the DAC and then a DNN improves overall performance. While this experiment clearly illustrates the DAC's ability to associate a particular feature with the abstention class, it might be argued the consistency of the smudge made this particular task easier than a typical real world setting; we provide a more challenging version of this experiment in the next section.

[4]We discuss abstention and memorization in Section 3.5

3.3.4 Results: Noisy Labels associated with a class

In this experiment, we simulate a scenario where a particular class, for some reason, is very prone to mislabeling, but it is assumed that given enough training data and clean labels, a deep network can learn the correct mapping. To simulate a rather extreme scenario, we randomize the labels over all the monkeys in the training set, which in fact include a variety of animals in the ape category (chimpanzees, macaques, baboons etc., Figure 3.3a) but all labeled as 'monkey'. Unlike the previous experiment, where the smudge was a relatively simple and consistent feature, the set of features that the DAC now has to learn are over a complex real-world object with more intra-class variance.

Detailed results are shown in Figure 3.3. The DAC abstains on most of the monkey images in the test set (Figure 3.3b), while abstaining on significantly fewer images in the other classes (Figure 3.3c), suggesting good abstention recall and precision respectively. In essence, the DAC has learned meaningful representation of monkeys, but due to label randomization, the abstention loss function now forces the DAC to associate monkey features with the abstention class. That is, the DAC, in the presence of label noise on this particular class, has learned a mapping from class features X to abstention class $K + 1$, much like a regular DNN would have learned a mapping from X to K_{monkey} in the absence of label noise. The representational power is unchanged from the DAC to the DNN; the difference is that the optimization induced by the loss function now redirects the mapping towards the abstention class.

Also shown is the performance of the baseline DNN in figures 3.3d to 3.3f. The prediction distribution over the monkey images spans the entire class range. That the DNN does get the classification correct about 20% of the time is not surprising, given that about 10% of the randomized monkey images did end up with the correct label, providing a consistent mapping from features to labels in these cases. However the accuracy on monkey images is poor; the distribution of the winning softmax scores over the monkey images for the DNN is shown in Figure 3.3e, revealing a high number of confident predictions ($p >= 0.9$) but closer

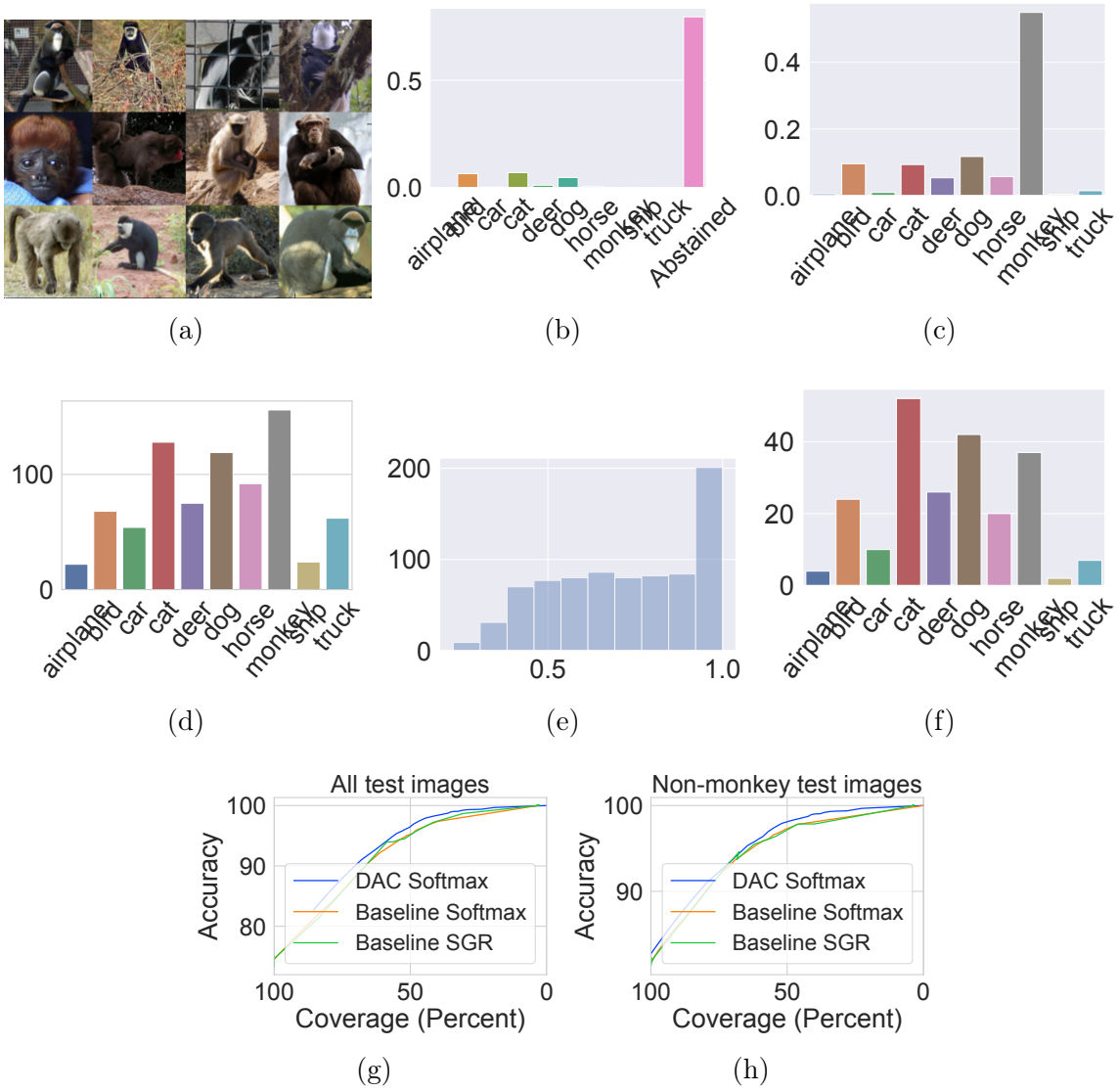


Figure 3.3: (a) All monkey images in the trainset had their labels randomized (b) The DAC abstains on about 80% of the monkey images (abstention precision). (c) Among images that were abstained, most of the images were those of monkeys (abstention precision). (d) Distribution of baseline DNN predictions over monkey images in the test set indicating learning difficulty on this class (e) Distribution of winning softmax scores of the baseline DNN on the monkey images (f) Distribution of baseline DNN softmax scores > 0.9. Most of these confident predictions are non-monkeys (g) Comparison against various abstention methods (softmax and SGR) on all the test images (h) Same comparison, but only on those images on which the DAC abstained. The DAC has a small but consistent advantage in both cases. All figures are computed on the test set.

inspection of the class distributions across just these confident predictions (3.3f) reveals that most of these predictions are incorrect suggesting that a threshold-based approach, which generally works well (Hendrycks and Gimpel, 2016; Geifman and El-Yaniv, 2017), will produce confident but erroneous predictions in such cases. This is reflected in the small but consistent risk-vs-coverage advantage of the DAC in Figure 3.3g and 3.3h. As before we compare both a softmax-thresholded DAC and baseline DNN, as well as the SGR method tuned on the baseline DNN scores. Unlike the random smudging experiment, here we do not eliminate the abstained images and retrain – doing so would completely eliminate one class. Instead we additionally compare the performance of the DAC on the images that it did not abstain (mostly non-monkeys), with the baselines (Figure 3.3h); the DAC has a small but significant lead in this case as well.

3.3.5 Results: Noisy Labels associated with a data transformation

We present further results on the abstaining ability of the DAC in the presence of structured noise. Here we simulate a scenario where a subset of the training data, due to feature degradation, ends up with unreliable labels. We apply a Gaussian blurring transformation to 20% of the train and test images across all the classes (Figure 3.4a), and randomize the labels on the blurred training set. This is similar to the smudging experiment, but here the conspicuous feature that the DAC can associate with abstention is the *absence of high frequency components*, or conversely, the abundance of low frequency components.

Results The DAC abstains remarkably well on the blurred images in the test set (Figure 3.4d), while maintaining classification accuracy over the remaining samples in the validation set ($\approx 79\%$). The baseline DNN accuracy drops to 63% (Figure 3.4b), while the basline accuracy over the blurred images alone is no better than random ($\approx 9.8\%$) . The abstention behavior of the DAC on the blurred images in the test set can be explained by how abstention evolves during training (Figure 3.4c). Once abstention is introduced at epoch 20, the DAC initially opts to abstain on a high percentage of the traning data, while continuing to learn (since the gradients w.r.t the true-class pre-activations are always negative.). In the

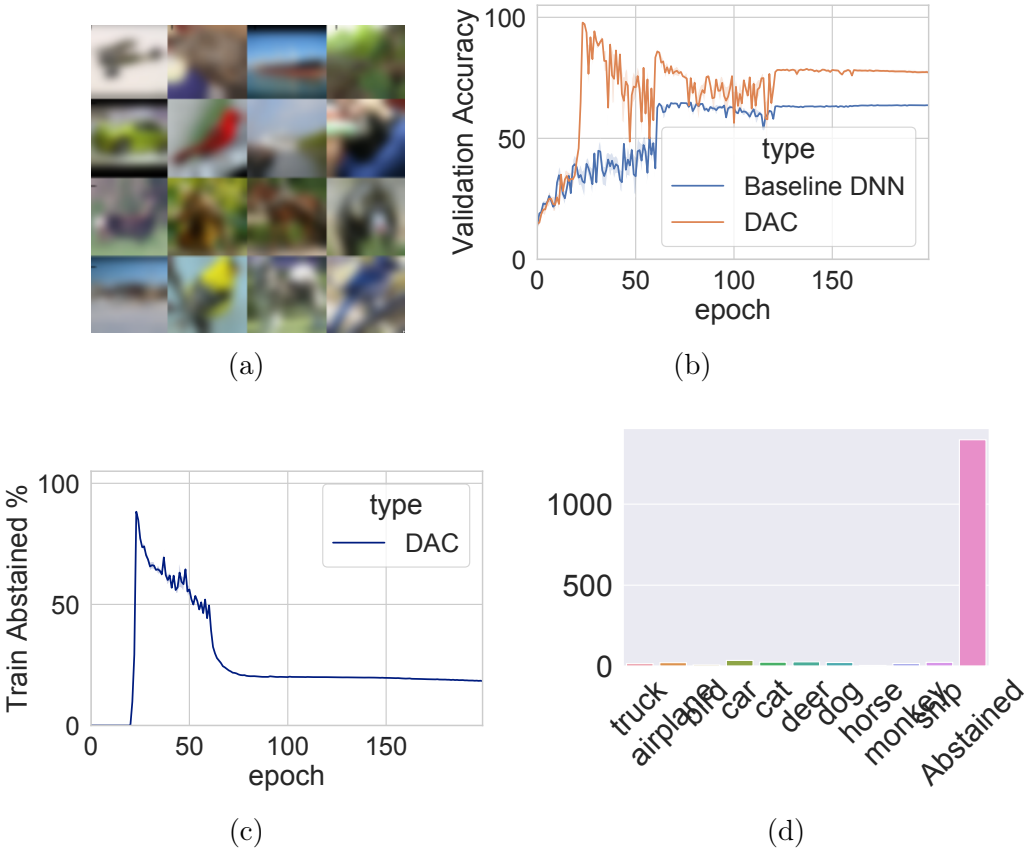


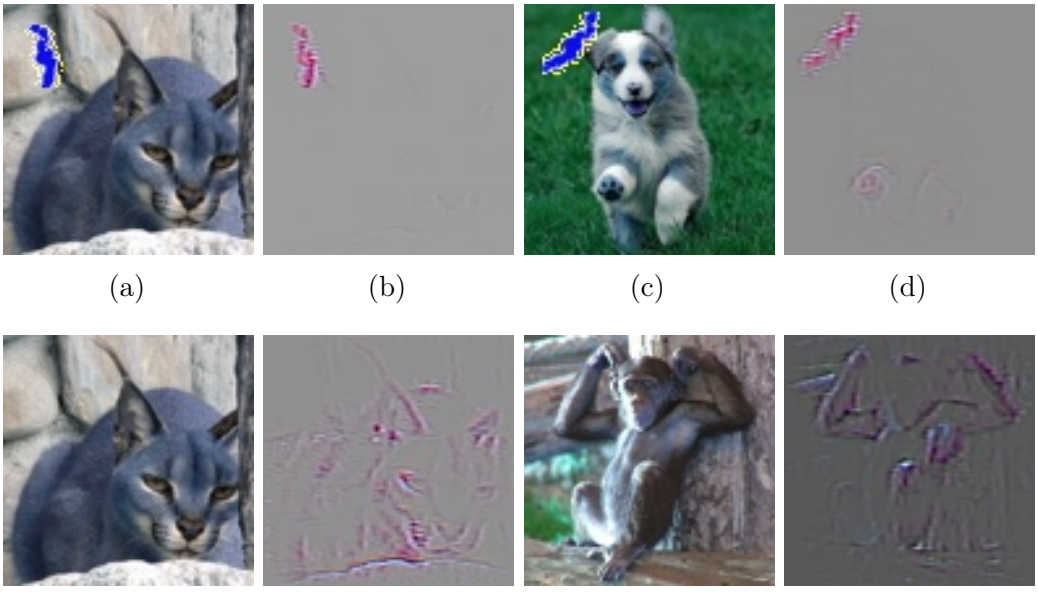
Figure 3.4: Results on blurred-image experiment with noisy labels (a)20% of the images are blurred in the train set, and their labels randomized (b) Validation accuracy for baseline vs DAC (non-abstained) (c)Abstention behavior for the DAC during training (d) Distribution of predictions on the blurred validation images for the DAC. We also observed (not shown) that for the baseline DNN, the accuracy on the blurred images in the validation set is no better than random.

later epochs, sufficient learning has taken place on the non-randomized samples but the DAC continues to abstain on about 20% of the training data, which corresponds to the blurred images indicating that a strong association has been made between blurring and abstention.

In summary, the experiments in this section indicate that the DAC can reliably pick up and abstain on samples where the noise is correlated with an underlying feature. In the next section, we peek into the network for better insights into the features that cause the DAC to

abstain.

3.3.6 What does the DAC "See"? Visual Explanations of Abstention



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.5: Filter visualizations for the DAC. When presented with a smudged image (a,c), the smudge completely dominates the feature saliency map (b,d) that cause the DAC to abstain. However for the same image without a smudge (e), the class features become much more salient (f) resulting in a correct prediction. For abstention on monkeys (g), the monkey features are picked up correctly (h), which leads to abstention

It is instructive to peer inside the network for explaining abstention behavior. Convolutional filter visualization techniques such as guided back-propagation (Springenberg et al., 2014) combined with class-based activation maps (Selvaraju et al., 2017) provide visually interpretable explanations of DNN predictions. In the case of the DAC, we visualize the final convolutional filters on the trained VGG-16 DAC model that successfully abstained on smudged and monkey images described in experiments in the previous section. Example visualizations using class-based activation maps on the predicted class are depicted in Fig-

ure 3.5. In the smudging experiments, when abstaining, the smudge completely dominates the rest of the features (Figures 3.5a through 3.5d) while the same image, when presented without a smudge (Figure 3.5e), is correctly predicted – with the actual class features being much more salient (3.5f) – implying that the abstention decision is driven by the presence of the smudge. For the randomized monkey experiment, while abstaining, it is precisely the features associated with the monkey class that result in abstention(Figures 3.5g, 3.5h), visually confirming our hypothesis in Section 3.3.4 that the DAC has effectively mapped monkey features to the abstention label.

3.4 Learning in the Presence of Unstructured Noise: The DAC as a Data Cleaner

So far we have seen the utility of the DAC in structured noise settings, where the DAC learns representations on which to abstain. Here we consider the problem of unstructured noise – noisy labels that might occur arbitrarily on some fraction of the data. Classification performance degrades in the presence of noise (Nettleton et al., 2010), with label noise shown to be more harmful than feature noise (Zhu and Wu, 2004). While there have been a number of works related to DNN training in the presence of noise (Sukhbaatar et al., 2014; Reed et al., 2014; Patrini et al., 2017), unlike these works we do not model the label flipping probabilities between classes in detail.

3.4.1 Approach

We simply assume that a fraction of labels have been uniformly corrupted and approach the problem from a data-cleaning perspective: using the abstention formulation and the extra class, can the DAC be used to identify noisy samples in the training set, with the goal of performing subsequent training, using a regular DNN, on the cleaner set? To identify the samples for elimination, we train the DAC, observing the performance of the non-abstaining part of the DAC on a validation set (which we assume to be clean). As mentioned before, this non-abstaining portion of the DAC is simply the DAC with the abstention mass normalized

out of the true classes. The result in Theorem 1 assures that learning continues on the true classes even in the presence of abstention. However at the point of best validation error, if there continues to be training error, then this is likely indicative of label noise; the loss function formulation makes the DAC more likely to abstain on such samples. It is these samples that are eliminated from the training set; we then re-train from scratch using regular cross-entropy loss on the non-eliminated samples[5].

To test the performance of the DAC, we compare two recent models that achieve state-of-the-art results in training with noisy labels on image data: MentorNet (Jiang et al., 2018b) that uses a data-driven curriculum-learning approach involving two neural nets – a learning network (StudentNet)and a supervisory network (MentorNet); and (Zhang and Sabuncu, 2018), that uses a noise-robust loss function formulation involving a generalization of the traditional categorical cross-entropy loss function.

3.4.2 Experimental Setup

We use the CIFAR-10, CIFAR-100 (Krizhevsky and Hinton, 2009) and the Fashion-MNIST (Xiao et al., 2017) datasets with an increasing fraction of arbitrarily randomized labels, using the same networks as the ones we compare to. In the DAC approach, both the DAC and the downstream DNN (that trains on the cleaner set) use the same network architectures. The downstream DNN is trained using the same hyperparameters, optimization algorithm and weight decay as the models we compare to. As a best-case model in the data-cleaning scenario, we also report the performance of a hypothetical oracle that has perfect information of the corrupted labels, and eliminates only those samples. To ensure approximately the same number of optimization steps as the comparisons when data has been eliminated, we appropriately lengthen the number of epochs and learning rate schedule for the downstream DNN (and do the same for the oracle.)

---

[5]Retraining from scratch gives significantly better results than continuing training on the non-abstained set, so we only report results using this protocol.

3.4.3 Results

Dataset	Method	Label Noise Fraction			
		0.2	0.4	0.6	0.8
CIFAR-10 (ResNet-34)	Baseline	88.94	85.35	79.74	67.17
	$\mathcal{L}_q$	89.83	87.13	82.54	64.07
	Trunc $\mathcal{L}_q$	89.7	87.62	82.7	67.92
	DAC	**92.91(0.24/0.01)**	**90.71(0.41/0.03)**	**86.30(0.56/0.07)**	**74.84(0.75/0.16)**
	Oracle	92.56(0.18)	90.95(0.36)	88.92(0.56)	86.43(0.72)
CIFAR-10 (Wide Res-Net 28x10)	Baseline	91.53	88.98	82.69	64.09
	MentorNet	92.0	89.0	-	49.0
	DAC	**93.35 (0.25/0.01)**	**90.93(0.43/0.01)**	**87.58(0.59/0.04)**	**70.8(0.77/0.17)**
	Oracle	95.17(0.18)	94.38(0.36)	92.74(0.56)	91.01(0.72)
CIFAR-100 (ResNet-34)	Baseline	69.15	62.94	55.39	29.5
	$\mathcal{L}_q$	66.81	61.77	53.16	29.16
	Trunc $\mathcal{L}_q$	67.61	62.64	54.04	29.60
	DAC	**73.55 (0.18/0.05)**	**66.92(0.25/0.01)**	**57.17(0.77/0.03)**	**32.16(0.87/0.33)**
	Oracle	77.15(0.2)	73.85(0.4)	69.48(0.6)	58.5(0.8)
CIFAR-100 (Wide Res-Net 28x10)	Baseline	71.24	65.24	57.56	30.43
	MentorNet	73.0	68.0	-	**35.0**
	DAC	**75.75(0.2/0.05)**	**68.2(0.57/0.01)**	**59.44(0.76/0.06)**	34.06(0.87/0.33)
	Oracle	78.76(0.2)	76.23(0.4)	72.11(0.6)	63.08(0.8)
Fashion-MNIST (ResNet-18)	Baseline	93.91	93.09	91.83	88.61
	$\mathcal{L}_q$	93.35	92.58	91.3	88.01
	$\mathcal{L}_q$	93.21	92.6	91.56	88.33
	DAC	**94.76(0.25/0.01)**	**94.09(0.48/0.01)**	**92.97(0.66/0.03)**	**90.79(0.88/0.04)**
	Oracle	95.22(0.18)	94.87(0.36)	94.64(0.56)	93.63(0.72)

Table 3.1: Comparison of performance of DAC against related work for data corrupted with uniform label-noise. The DAC is used to first filter out noisy samples from the training set and a DNN is then trained on the cleaner set. Each set also shows the performance of the baseline DNN trained without removing noisy data. Also shown is the performance of a hypothetical oracle data-cleaner that has perfect information about noisy labels. The numbers in parentheses next to the DAC indicate the fraction of training data filtered out by the DAC and the remaining noise level. For the oracle, we report just the amount of filtered data (remaining noise level being zero). $\mathcal{L}_q$ and truncated $\mathcal{L}_q$ results reproduced from (Zhang and Sabuncu, 2018); MentorNet results reproduced from (Jiang et al., 2018b)

Results are shown in Table 3.1. By identifying and eliminating noisy samples using the DAC and then training using the cleaner set, noticeable – and often significant – performance improvement is achieved over the comparison methods in most cases. Interestingly, in the case of higher label randomization, for the more challenging data set like CIFAR-10 and CIFAR-100, we see the noisy baseline outperforming some of the comparison methods. The

DAC is however, consistently better than the baseline. On CIFAR-100, for 80% randomization, the other methods often have very similar performance to the DAC. This is possibly due to the substantial amount of data that has been eliminated by the DAC leaving very few samples per class. The fact that the performance is comparable even in this case, and the high hypothetical performance of the oracle illustrate the effectiveness of a data cleaning approach for deep learning even when a significant fraction of the data has been eliminated.

3.4.4 Comparison to Related Work

While data cleaning (or pruning) approaches have been considered before in the context of shallow classifiers (Angelova et al., 2005; Brodley and Friedl, 1999; Zhu et al., 2003), to the best of our knowledge, this is the first work to show how abstention training can be used to identify and eliminate noisy labels for improving classification performance. Besides the improvements over the work we compare to, this approach also has additional advantages: we do not need to estimate the label confusion matrix as in (Sukhbaatar et al., 2014; Reed et al., 2014; Patrini et al., 2017) or make assumptions regarding the amount of label noise or the existence of a trusted or clean data set as done in (Hendrycks et al., 2018b) and (Li et al., 2017b).

The DAC approach is also significantly simpler than the methods based on the mentor-student networks in (Jiang et al., 2018b) and (Han et al., 2018), or the graphical model approach discussed in (Vahdat, 2017). In summary, the results in this section not only demonstrate the performance benefit of a data-cleaning approach for robust deep learning in the presence of significant label noise, but also the utility of the abstaining classifier as a very effective way to clean such noise.

3.4.5 Results on Non-Uniform Label Noise

Often in real-world datasets, label noise is not completely arbitrary, but instead, visually or semantically similar classes are often confused with each other. Unlike the structured label noise experiments, where a corrupting feature or transformation resulted in corrupted labels,

Dataset	Method	Class Dependent Label Noise Fraction			
		η =0.1	0.2	0.3	0.4
CIFAR-10 (ResNet-34)	$\mathcal{L}_q$	90.91	89.33	85.45	76.74
	Trunc $\mathcal{L}_q$	90.43	89.45	87.10	82.28
	Forward T	91.32	90.35	89.25	88.12
	Forward $\hat{T}$	90.52	89.09	86.79	83.55
	DAC	**94.23**	**93.20**	**92.07**	**89.88**
CIFAR-100 (ResNet-34)	$\mathcal{L}_q$	68.36	66.59	61.45	47.22
	Trunc $\mathcal{L}_q$	68.86	66.59	61.87	47.66
	Forward T	71.05	71.08	70.76	**70.82**
	Forward $\hat{T}$	45.96	42.46	38.13	34.44
	DAC	**75.59**	**73.22**	**71.38**	65.34
Fashion-MNIST (ResNet-18)	$\mathcal{L}_q$	93.51	93.24	92.21	89.53
	Trunc $\mathcal{L}_q$	93.53	93.36	92.76	91.62
	Forward T	94.33	94.03	93.91	93.65
	Forward $\hat{T}$	94.09	93.66	93.52	88.53
	DAC	**95.48**	**95.08**	**94.96**	**94.31**

Table 3.2: Comparison of DAC vs related methods for class-dependent label noise. Performance numbers reproduced from (Zhang and Sabuncu, 2018). For the DAC, an abstaining classifier is first used to identify and eliminate label noise, and an identical DNN is then used for downstream training.

here we model class dependent label noise by assigning a probablity η that class i is labeled as class j. We report results on CIFAR-10, CIFAR-100 and Fashion-MNIST for using the experimental setup as described in (Zhang and Sabuncu, 2018), and we compare with the results reported in that paper which also includes the Forward correction method of (Patrini et al., 2017). In CIFAR-10, the class dependent noise results in the following flip scenario with probability η: TRUCK $\rightarrow$ AUTOMOBILE, BIRD $\rightarrow$ AIRPLANE, DEER $\rightarrow$ HORSE, and CAT $\leftrightarrow$ DOG. For CIFAR-100, classes are organized into groups as described in (Krizhevsky and Hinton, 2009) and the class-dependent noise is simulated by flipping each class into the next circularly with probability η. For Fashion-MNIST, classes are flipped as follows: BOOT $\rightarrow$ SNEAKER , SNEAKER $\rightarrow$ SANDALS, PULLOVER $\rightarrow$ SHIRT, COAT$\rightarrow$ DRESS with

probability η. The aforementioned flipping scenarios are identical to the setup in (Zhang and Sabuncu, 2018), even though it must be noted that the likelihood of label flips among similar classes generally depends on underlying features in a given sample. However, existing work only models class-conditional label noise(i.e. without taking features into account), and thus we use the same assumptions here[6]. Results are shown in Table 3.2, that illustrate the ability of the DAC as an effective data-cleaner even nder non-uniform, class-conditional label noise scenarios.

3.5 Abstention and Memorization

In the structured-noise experiments in Section 3.3, we saw that the DAC abstains, often with near perfection, on label-randomized samples by learning common features that are present in these samples. However, there has been a considerable body of recent work that shows that DNNs are also perfectly capable of memorizing random labels (Zhang et al., 2016; Arpit et al., 2017). In this regard, abstention appears to counter the tendency to memorize data; however it does not generally prevent memorization.

In the experiments described in Section 3.3.3, the desired behavior of abstaining on the smudged samples (whose labels were randomized in the training set) does not persist indefinitely. At epochs 60 and 120, there are steep reductions in the abstention rate, coinciding with learning rate decay. It appears that at this point, the DAC moves into a memorization phase, finding more complex decision boundaries to fit the random labels, as the lower learning rate enables it to descend into a possibly narrow minima. Generalization performance also suffers once this phase begins. This behavior is consistent with the discussion in (Arpit et al., 2017) – the DAC does indeed first learn patterns before literally descending into memorization.

Auto-tuning of α described in Section 3.2.1 prevents abstention rate from saturating to 1; however as we saw in Section 3.3.3, it does not prevent abstention from converging to 0. A

[6]The ability to learns features that are correlated with label noise, is of course, one of the unique properties of the DAC, as demonstrated in the various experiments in Section 3.3.

sufficiently small learning rate and long training schedule eventually results in memorization. As discussed in the previous section, to prevent this, we employ early stopping when a desirable validation error has been reached on the non-abstaining part of the DAC.

3.6 Conclusions

There are few works discussing abstention in the context of deep learning and to the best of our knowledge, we are the first to show its effectiveness while training. We showed the utility of the DAC under multiple types of label noise: as a representation learner in the presence of structured noise and as an effective data cleaner in the presence of arbitrary noise. Results indicate that data-cleaning with the DAC significantly improves classification performance for downstream training. Furthermore, the loss function formulation is simple to implement and can work with any existing DNN architecture; this makes the DAC a very useful addition to real-world deep learning pipelines.

Chapter 4

EXTENSIONS TO THE DEEP ABSTAINING CLASSIFIER

In this chapter, we describe various extensions to the deep abstaining classifier introduced in Chapter 3. Building on previous results, we demonstrate the use of *PID control* to improve the performance of the DAC. Next, we augment the fully supervised training approach described so far with *semi-supervised learning* for further gains on label-noise robustness.

4.1 Stabilizing Abstention Behavior of the DAC

Consider again the loss on a sample x, introduced in Section 3.1:

$$\mathcal{L}(x) = (1 - p_{k+1}) \left(- \sum_{i=1}^{k} t_i \log \frac{p_i}{1 - p_{k+1}} \right) + \alpha \log \frac{1}{1 - p_{k+1}} \tag{4.1}$$

The hyperparameter α, which is applied to the abstention penalty term, controls the abstention behavior. We established, analytically, in Section 3.2 the condition for the DAC to increase mass in the abstention class to be:

$$\alpha < (1 - p_{k+1}) \left(- \log \frac{p_j}{1 - p_{k+1}} \right) \tag{4.2}$$

where j is the class specified in the training data for a given sample x (which may not be the true class due to label noise). In the linear ramping-up of α discussed in the previous chapter, the above condition will fail to hold at some point during the training, and as we have seen, abstention eventually decays to zero. In this scenario, the ideal rate of abstention was determined by observing the performance on a clean validation set. Often, however, a sufficiently large clean validation set might not be available since cleaning is labor intensive. In these situations, it might be easier to determine an approximate estimate of label noise; the

desired behavior for the DAC would be then to stabilize abstention to match the estimated noise rate.

Note that p_j in the above condition is a moving target: as established in Section 3.2, as long as $\alpha > 0$, the DAC continues to learn on the training data implying all the mass will eventually end up in p_j. If at some iteration t, the ideal level of abstention has been reached, then to maintain abstention at that rate, α has to be dynamically adjusted during training. The ramping-up of α described in Algorithm 1 does not result in this behavior; on the contrary, it was purposely designed to *decay* abstention such that the DAC abstains on only the most confusing samples in the later stages of training. Here, we need to *maintain abstention* once it has reached the noise rate by *reducing* α. But by how much? As we have seen, abstention is very sensitive to α – reduce too much and abstention can quickly shoot up, while an insufficient reduction can end up causing the DAC to switch to non-abstaining behavior (Figure 4.1)

The mechanism for dynamically adjusting α – our control knob– to stabilize abstention behavior of the DAC – our "process" – falls in the realm of classical control, and that is what we explore in this chapter. Specifically, we explore *proportional-integrative-derivative* (PID) control – a widely used technique in industrial processes – to tune α. Note that even though α is a hyper-parameter – in the sense that it is not a learned parameter – we can tune it outside the learning loop by observing its behavior on the training set. In that sense, α is very much like the learning rate in SGD, which is also dynamically annealed in most implementations. There is a subtle difference however: a good learning rate annealing schedule is generally determined by observing the performance of the DNN on a validation set; in contrast, our aim here is to achieve the desired abstention behavior on the *training set* and thus we do not need a clean validation set to determine the correct α; this is especially useful in situations when a clean validation set is not available.

Before we proceed, a note on abstention set-point: to effectively use a control scheme to stabilize to a set-point, one obviously needs to determine the set-point, which in our case, is the estimated label noise rate. In Section 4.5 we show how by employing classical statistics

this can be determined in a practical manner by looking at only a very small fraction of the training data. In the following sections, we give an overview of PID control, make connections to related work, and empirically demonstrate its effectiveness.

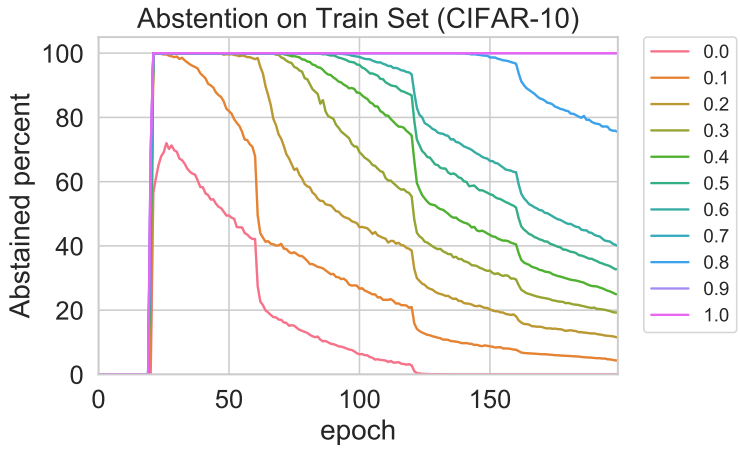


Figure 4.1: Abstention on train set at various levels of label randomization (indicated by color). When α (not shown), is linearly ramped up, the DAC starts overfitting and abstention eventually decays to zero.

4.2 An Overview of PID Control

PID control is a form of continuous feedback control based on the idea that the control signal should be proportional to the behavior of the error signal: at a given time step t, we use the observed error, past error and how quickly the error is changing to decide the control signal for the next time step. Devices using the idea of continuous feedback control date back to the 17th century with early theoretical work by Maxwell in the nineteenth century, and a precise mathematical formulation appearing in the early 1920's (Bennett, 1993).

Formally, in a PID system, the control signal $u(t)$ is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \tag{4.3}$$

where $e(t)$ is the error at time t, defined as the difference between observed value and set-point.

The integral term gives the accumulation of past errors, while the derivative term indicates how fast the error is changing in response to the previous control signal; the coefficients K_p, K_i and K_d determine the contribution of each of these terms. This simple scheme allows one to treat the system being controlled using a model-free, "black-box " approach just by observing the output, which is a very convenient situation for us because modern DNNs are highly non-linear systems involving hundreds of millions of parameters. Even though we have an analytical condition on abstention behavior, it is generally not possible to know in advance how the DAC – which is just a DNN with a modified loss function – will respond to changes in α. PID control is surprisingly robust and effective: over 90% of industrial contrllers are implemented based on PID (Ang et al., 2005) including applications in aeronautics (Salih et al., 2010), self-driving cars (Zhao et al., 2012) and robotics (Rocco, 1996).

4.2.1 PID Control in Deep Learning

While PID control has not been explicitly used in deep learning, it is a remarkable fact that gradient descent – where the parameters are updated based on the gradient of the loss – *is a form of PID*. In fact, plain stochastic gradient descent (without momentum) is a proportional-only controller, since the parameter update rule for SGD

$$\Theta_{t+1} = \Theta_t - \eta \frac{\partial \mathcal{L}_t}{\partial \Theta_t} \tag{4.4}$$

can be compared to a proportional controller by treating the gradient $\frac{\partial L_t}{\partial \Theta_t}$ as the error and the negative learning rate η as the proportional constant K_p. Further, SGD with momentum, where the update rules are

$$V_{t+1} = \alpha V_t - \eta \frac{\partial L_t}{\partial \Theta_t} \tag{4.5}$$

$$\Theta_{t+1} = \Theta_t + V_{t+1} \tag{4.6}$$

can, after some mathematical manipulation, be written as

$$\Theta_{t+1} = \Theta_t - \eta \frac{\partial L_t}{\partial \Theta_t} - r \left(\sum_{i=0}^{t-1} \left(\partial L_i / \partial \theta_i \alpha^{t-i} \right) \right) \tag{4.7}$$

which is equivalent to a Proportional-Integral (PI) controller. This connection to SGD was recently made in (An et al., 2018) where the authors used a PID-based approach to dynamically tune the learning rate of SGD, resulting in improved convergence in both training and validation losses on various image datasets. This is one of the few works, and perhaps the first, to apply full-fledged PID control for optimization in deep learning.

It has also been argued that learning to learn – or *meta-learning* (Vanschoren, 2018) – is a special case of controller design (Recht, 2018). In meta-learning, the current dominant paradigm appears to be reinforcement learning (Sutton et al., 1998), which is of course a much more sophisticated approach to policy learning, and is used in more complicated situations where simple PID-type control approaches will not suffice. Traditionally, PID control been applied only in the realm of low-level control loops; nevertheless, the complexity of DNN behavior not-withstanding, the abstention dynamics established in the previous chapter allows us to treat the problem of stabilizing abstention as involving one control knob and avoid more complicated schemes like reinforcement learning.

4.3 Stabilizing Abstention with PID Control

As previously discussed, to maintain and stabilize abstention at a given set-point, the penalty on abstention has to be continuously reduced; otherwise, the DAC attempts to minimize loss by over-fitting to the mislabeled data. To do this, we observe abstention rate at each learning iteration and suitably anneal α using the PID approach.

Consider again the PID control rule:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \tag{4.8}$$

Since in our case, the control is implemented on a digital computer, we need the discrete-time version of this. Let η be the desired abstention rate, and r_t be the abstention at time-step t. Then the error term in the discrete time of version of the PID becomes:

$$e_t = \eta - r_t \tag{4.9}$$

$$\Delta e_t = e_t - e_{t-1} \tag{4.10}$$

$$= r_{t-1} - r_t \tag{4.11}$$

We compute the integral term by maintaining a cumulative sum of the errors, scaled by a suitable time constant

$$w_t = w_{t-1} + e_t \Delta t \tag{4.12}$$

The derivative is calculated using a finite difference:

$$v_t = \frac{\Delta e_t}{\Delta t} \tag{4.13}$$

Putting everything together, the control signal of the discrete-time PID becomes:

$$u_t = K_p e_t + K_i w_t + K_d v_t \tag{4.14}$$

where K_p, K_i and K_d are the non-negative PID constants. This control signal is then used to update α as follows:

$$\alpha_t = max(0, \alpha_{t-1} - u_t) \tag{4.15}$$

Note that α is *reduced* if the control signal is positive. This can be understood as follows: if e_t and w_t are positive, then abstention is below the setpoint and thus α has to be reduced to encourage abstention. If v_t is positive, then the abstention is on a decreasing trajectory, so

Algorithm 2: An optimization step in the DAC with PID tuning of α

Input: Predictions on current batch ($\mathbf{P} \in \mathbb{R}^{|\mathbf{B}| \times (\mathbf{K+1})}$, where $|B|$ is batch-size)
$\hat{r}_t \leftarrow \frac{1}{|B|} \sum_{1 \leq i \leq |B|} \mathbb{1}\left(\text{argmax}\left(\mathbf{P}_{\mathbf{i},:}\right) = K+1\right)$ // `instantaneous abstention rate`
$r_t \leftarrow \mu r_{t-1} + (1-\mu)\hat{r}_t$ // `smoothed abstention rate`
$u_t \leftarrow$ PID (r_t) // `compute control using Equations 4.9 through 4.14`
$\alpha_t \leftarrow \max(0, \alpha_{t-1} - u_t)$ //α `update step. `$\alpha \geq 0$` is required for learning. See Section 3.2`
$\mathcal{L}_t \leftarrow$ DAC_LOSS $(\mathbf{P}, \alpha_\mathbf{t})$ //`compute loss using Equation 3.1`
SGD_UPDATE$(\mathcal{L}_t)$ // `Gradient descent update based on current loss`

to damp the correction, we decrease α. The minimum value of α has to be clamped at zero to ensure that learning takes place on the true class (See Section 3.2). Also note that since we are performing stochastic optimization via SGD, the instantaneous abstention rate $\hat{r}_t$, calculated on a mini-batch, is a noisy measurement of the true abstention rate. To smooth out the effect of this noise, we maintain an exponentially weighted moving average of abstention as follows:

$$r_t = \mu r_{t-1} + (1-\mu)\hat{r}_t \tag{4.16}$$

where $\hat{r}_t$ is the instantaneous abstention rate calculated on the current SGD batch. A pseudo-code of the implementation is given in Algorithm 2 which shows how the PID tuning is incorporated into the optimization loop.

4.4 Experiments

Experimental results using the PID approach are discussed in this section. Our setup is similar to the one we used in Section 3.3.2. As before, we test our method for both structured and unstructured noise. Since we do not use the linear ramping of α, we do not have to worry about the hyperparameters involved in initializing and ramping α. However, we do have to set the PID tuning coefficients. We manually tune these using the Ziegler-Nichols rules (Ziegler

and Nichols, 1942) as a guide. We found that the value of $K_p = K_i = 0.1, K_d = 0.05$ worked well, and we use that in all our experiments.

4.4.1 Structured Noise

In the first set of experiments, we test the PID controlled DAC on the corrupted version of the STL-10 dataset for the random smudge, random monkeys and random blurred described in Section 3.3.2. We use the VGG-16 architecture as before, and the same loss function, but this time with the PID tuning loop. Abstention set point is set to the label-noise rate, which is 10% of the data in each of the three cases.

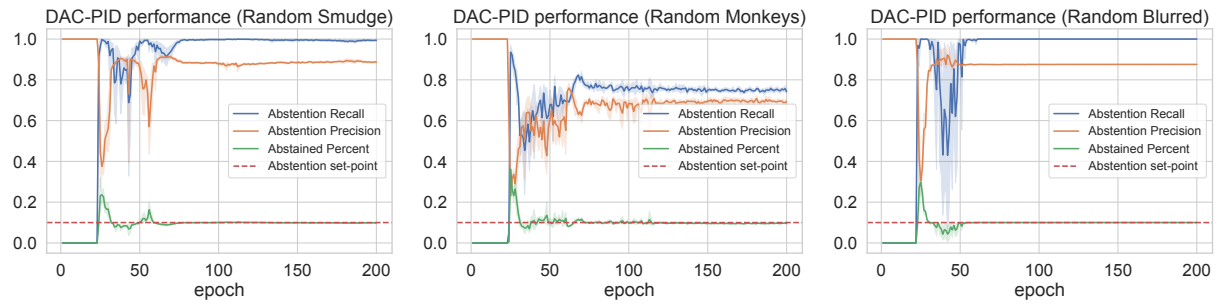


Figure 4.2: Performance of a PID-controlled deep abstaining classifier on various datasets with structured noise, in terms of abstention stablization to set-point (dotted red line) and the quality of abstention (precision and recall).

Results are shown in Figure 4.2. For the random smudge, and random blurring experiments, the abstention is very accurate, both in terms of precision and recall. On the random monkey set, abstention quality is lower but no worse than the best performing case in the non-PID version of DAC, meaning we have achieved the same quality of abstention as before without having to worry about when to stop training. The abstention rate converges to the set-point in about 60 epochs which is when the learning rate is annealed, following which the behavior is remarkably stable.

4.4.2 Unstructured Noise

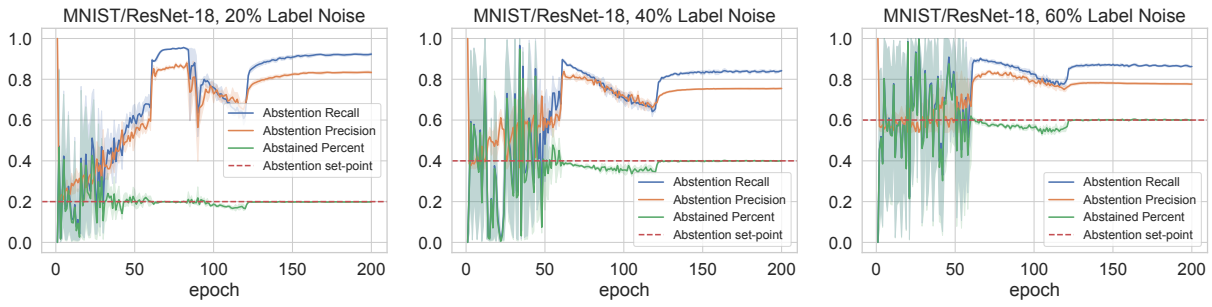


Figure 4.3: Performance of a PID-controlled deep abstaining classifier on the MNIST dataset on ResNet-18 architecture, at various levels of label noise. Shown are abstention stabilization to set-point and the quality of abstention (precision and recall).

In this section, as in Section 3.4.2, we perform experiments the on the MNIST, CIFAR-10 and CIFAR-100 datasets with an increasing fraction of uniformly randomized labels (symmetric noise model), using the same networks as before: an 18 and 34 layer residual network, and 28×10 wide residual network. Unlike the experiments in Section 3.4.2, here we are only interested in the stabilization and quality of abstention, so do not perform downstream experiments with the cleaner dataset.

Results on the various datasets are shown in Figures 4.3 and 4.4. Since MNIST is an easier dataset, the abstention precision and recall are slightly better than the more challenging cases of CIFAR-10 and CIFAR-100. The quality of stabilization dynamics is remarkably similar across datasets, and architectures: in all cases, abstention takes longer to stabilize with increasing amount of label noise. This is not surprising, given that a large fraction of corrupted data – whose labels have been uniformy randomized – cause a significant amount of variance in the gradient update in each iteration. It has been observed that fitting to the training data indeed takes longer in the presence of severe label noise (Zhang et al., 2016); the fact that abstention also takes longer to stabilize is consistent with this.

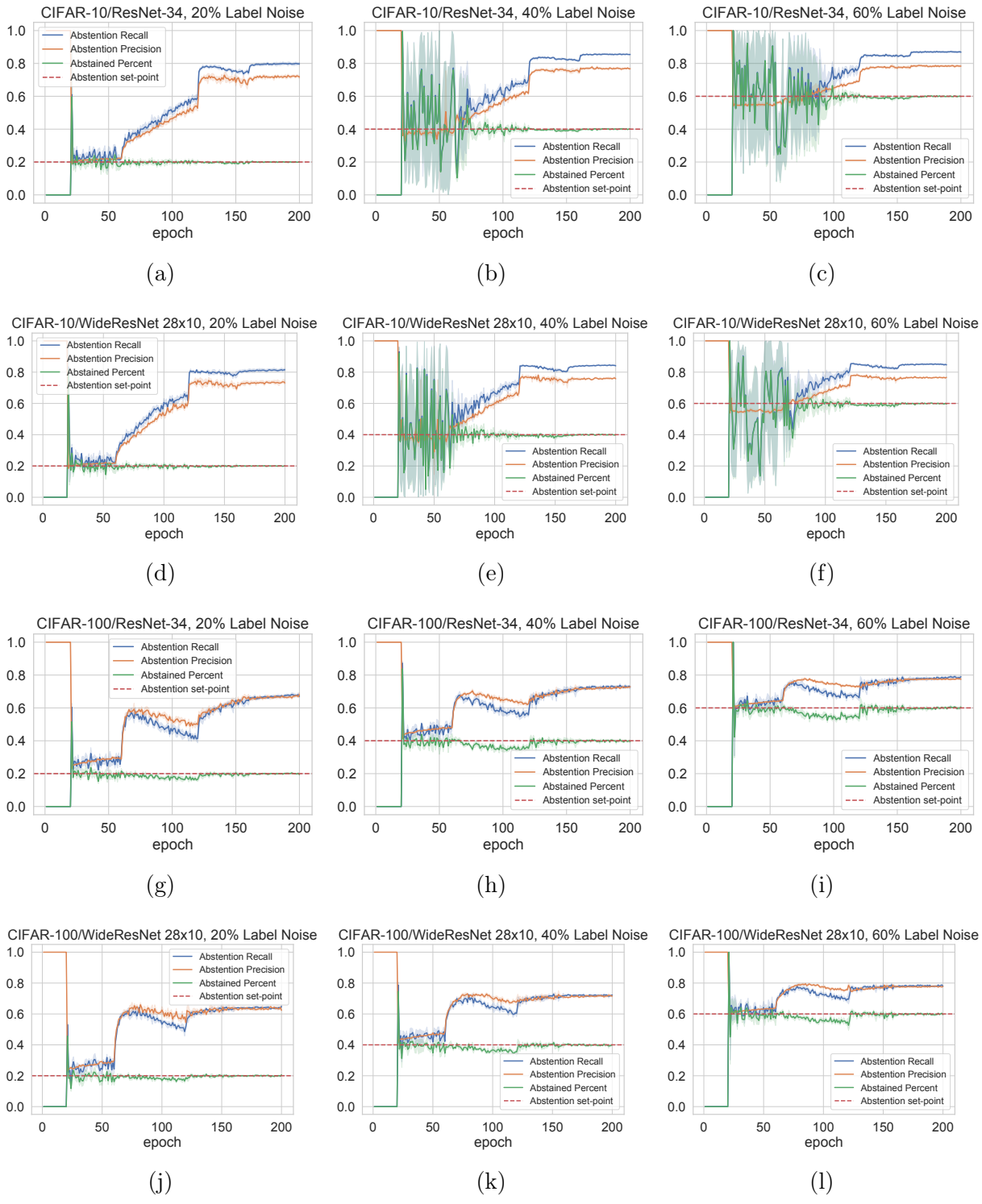


Figure 4.4: Performance of a PID-controlled deep abstaining classifier on the CIFAR-10 and CIFAR-100 on two different architectures, at various levels of label noise. Shown are abstention stabilization to set-point and the quality of abstention (precision and recall).

It is worthwhile noting that at high label noise rates (60%), even though the PID-controlled DAC achieves a precision of 80%, a significant amount of label noise still remains in the non-abstained data. Further downstream training will require noise-robust methods to train a performant classifier; we explore such methods in the second half of this chapter where we discuss semi-supervised learning.

4.5 Setting the Setpoint: Estimating Label Noise via Sampling

While the experiments in the previous section showed that a PID-controlled DAC can be used to stabilize abstention behavior and identify label noise both with high precision and high recall, the abstention rate – which in the experiments was set to the label noise rate – has to be specified by the user, meaning an estimate of the label noise is needed a priori. How do we reliably determine the amount of label noise in a dataset, especially when the number of training samples can be in the millions? One approach to this is to train the model on the corrupted set $\tilde{\mathcal{D}}$, and then assuming that the model has learnt the conditional noisy distribution $p(\tilde{y}|x)$, we can use the model's predictions on a clean set $\mathcal{D}$ to estimate label noise. The idea here is that prediction errors on $\mathcal{D}$ will be indicative of the label corruptions in the training set, and thus allow us to estimate the label transition matrix $Q_{i,j}$, and hence, also the total noise rate. This method was first described in (Patrini et al., 2017) and a variant of this approach was recently proposed in (Hendrycks et al., 2018b).

Note that for estimating the abstention setpoint (the total label noise rate), we do not actually need the full label transition matrix Q, and further, if a clean validation set is not available, then the above-mentioned methods are not applicable. In such cases one can estimate the label noise by manual sampling of the data. But how big a sample size do we need? It turns out that due to the Central Limit Theorem, we need a surprisingly small number of samples – only in the few hundreds – to reliably estimate label noise in the training set even for very large datasets.

To estimate label noise, we formulate the problem as estimating the proportion of successes, p, in a dichotomous i.e., binary -outcome variable in a population, where a success here

is defined as the event that a data sample is incorrectly labeled. From the Central Limit Tteorem, we know that the distribution of the sample means is approximately normally distributed (provided the sample size is not too small) with a mean and variance given by:

$$\mu_{\bar{X}} = p \tag{4.17}$$

$$\sigma^2_{\bar{X}} = \sigma^2/n \tag{4.18}$$

where $\bar{X}$ is the sample, p is the proportion we are trying to estimate and σ^2 is the true variance of the population. For a dichotomous random variable, $\sigma^2 = p(1-p)$. If we denote $\hat{p}$ to be the sample mean, we have

$$\hat{p} \sim \mathcal{N}(p, p(1-p)/n) \tag{4.19}$$

Let z represent the standard normal variable corresponding to $\hat{p}$, that is:

$$z = \frac{\hat{p} - p}{\sqrt{\frac{p(1-p)}{n}}} \tag{4.20}$$

Also, for the standard normal distribution, we have:

$$P(-z_{\alpha/2} < z < z_{\alpha/2}) = 1 - \alpha \tag{4.21}$$

that is, with a probability of $1-\alpha$, z lies in the interval $[-z_{\alpha/2}, z_{\alpha/2}]$. Concretely, for $\alpha = 0.05$ (corresponding to a confidence level of 95%), we have $z_{\alpha/2} = 1.96$, giving

$$P(-1.96 < z < 1.96) = 0.95 \tag{4.22}$$

Substituting the value of z, we have

$$P(-1.96 < \frac{\hat{p} - p}{\sigma/\sqrt{n}} < 1.96) = 0.95 \tag{4.23}$$

where $\sigma = p(1-p)$. Re-arranging, and after a little algebra, we have

$$P(\hat{p} - 1.96\frac{\sqrt{\sigma}}{n} < p < \hat{p} + 1.96\frac{\sqrt{\sigma}}{n}) = 0.95 \tag{4.24}$$

The quantity $1.96\sigma/\sqrt{n}$ is the margin of error at the 95% confidence level. For example, if we wanted the margin of error to be no more than ϵ at a 95% confidence interval, we have

$$1.96\sigma/\sqrt{n} \leq \epsilon \tag{4.25}$$

Substituting $\sigma = p(1-p)$ and re-arranging gives us the lower bound on sample size for the required margin of error and confidence level of 95%.

$$n \geq \left(\frac{1.96}{\epsilon}\right)^2 p(1-p) \tag{4.26}$$

Note that this involves the quantity p we are trying to estimate. However, the expressions $p(1-p)$ is maximized when $p = 0.5$, and thus we arrive at the worst-case (or most conservative) minimum sample size for our required margin of error ϵ and confidence level.

$$n \geq \frac{0.96}{\epsilon} \tag{4.27}$$

Concretely, for a margin of error of $\pm 5\%$, this works out to a minimum sample size of ≈ 380 samples, meaning label noise can be estimated with high confidence by manually examining only a few hundred samples. While this is not trivial, it is feasible in a real-worldlmachine learning setting. Note that the minimum sample size is essentially independent of the population size[1], meaning a few hundred samples will suffice for label noise estimation even when the training data contains millions of samples.

[1]The correction factor of $\sqrt{\frac{N-n}{N-1}}$ can typically be ignored unless the sample size n is comparable to the population size N

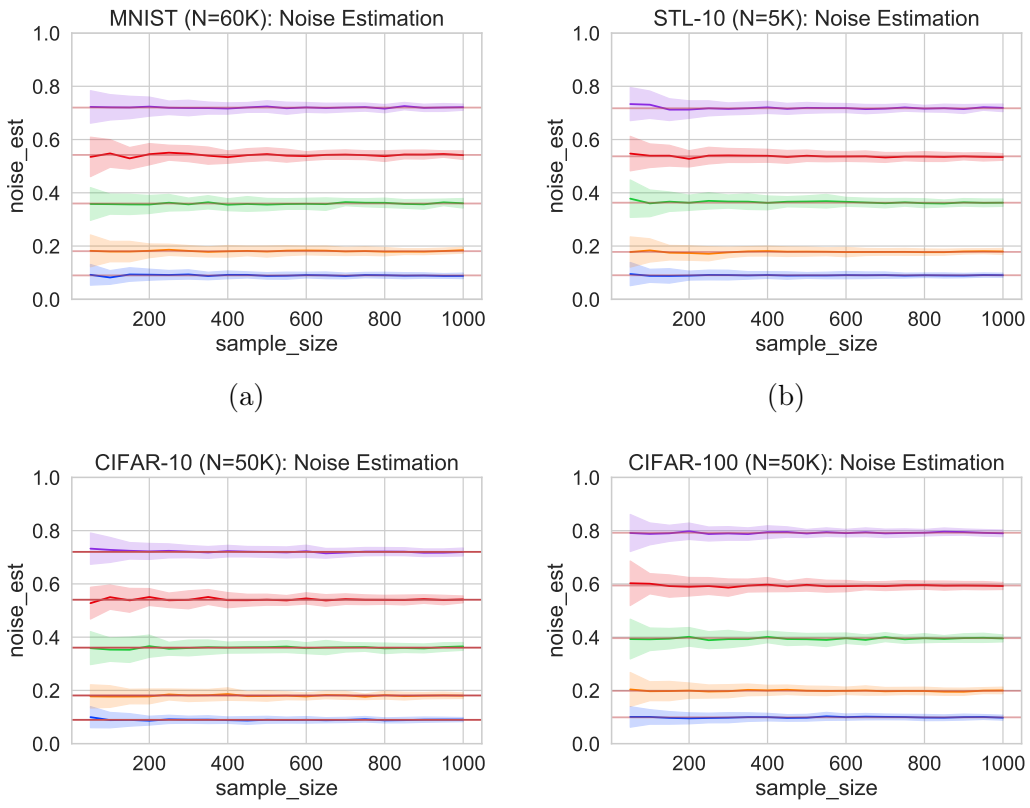


Figure 4.5: Estimated label noise at various sample sizes and at different noise rates on various image datasets. MNIST, STL-10 and CIFAR-10 have objects from ten different categories, while CIFAR-100 consists of one hundred categories.

In Figure 4.5, we show the results of a few label noise simulation experiments over various image datasets, at various levels of label noise. The solid red horizontal line indicates the true label noise rate, while the colored bands indicate the estimated label noise and variance of the estimate at a given sample size. As expected from the earlier analysis, a relatively small number of samples suffices to reliably estimate label noise, and this is independent of the number of classes or the population size.

4.6 Discussion

This concludes the first half of this chapter on stabilizing abstention in the DAC. We see that using PID control, abstention can be effectively stabilized at a given setpoint, to identify mislabeled data both with good precision and recall. Using the Central Limit Tteorem and sampling, we were able to estimate the approximate noise rate by only examining a few hundred training samples; this allows us to set the abstention set-point to a value that is very near the true label noise rate.

More sophisticated methods such as reinforcement learning could possibly be used. However, the analytical results developed in Chapter 1 allow us to treat this a relatively simple control system with one input (α), – aone output (– absteon rate). This simplicity and the demonstrated effectiveness of the PID approach makes this an appealing choice for tuning the DAC in real-world scenarios.

4.7 Semi-Supervised Deep Learning with the Deep Abstaining Classifier

While the abstention based approach described so far is useful for eliminating noisily labeled data, large quantities of noisy labels might result in a significant fraction or even the majority of training data being eliminated. However, this prevents us from effectively utilizing DNNs which require a large amount of training data to perform well. If instead, we were to just ignore the noisy labels, but still make use of the features for training, we could potentially further improve performance. This is the realm of semi-supervised learning (SSL), where both labeled and unlabeled data are used to improve classification performance (Chapelle et al., 2006). Originally developed in the context of shallow learning models, there has been a surge of interest in SSL in the last decade especially in domains where data has been plentiful such as images, text, speech and bio-informatics (Chapelle et al., 2006). The field of SSL is vast and uses a diverse set of tools, and a full discussion is beyond the scope of this thesis, but below we give an overview of various paradigms that have been successfully used to leverage the power of unlabeled data.

Concretely, let $\{(\mathbf{x_i}, \mathbf{y_i})\}_{\mathbf{i=1}}^{\ell}$ be the labeled training data and $\{\mathbf{x_i}\}_{\mathbf{i=\ell+1}}^{\ell+\mathbf{u}}$ be the unlabeled training data, where $n = \ell + u$ so that we have n points in total. In supervised learning, the goal is to learn a mapping from $\mathbf{x}$ to $\mathbf{y}$ while in unsupervised learning, the task is fundamentally that of estimating the density $p(x)$. Semi-supervised learning is halfway between these two, where unlabeled data is leveraged in a specific way with the intention of producing a better mapping from $\mathbf{x}$ to $\mathbf{y}$, since the task is generally still that of learning a classifier.

However, for SSL to work, the additional knowledge gained about $p(x)$ through unlabeled data has to be somehow useful for the inference of $p(y|x)$; otherwise, SSL will not yield any improvement over purely supervised learning and might even degrade performance. Concretely, SSL provides a tangible benefit when the following conditions are met:

- **Manifold Assumption** Data are embedded in a low-dimensional non-linear manifold in some high dimensional ambient space

- **Smoothness Assumption** Nearby points are likely to have same labels

- **Cluster Assumption** Points in same class are likely to cluster in high density regions

Fortunately, in the vast majority of situations, these assumptions roughly hold (Figure 4.6) and it is indeed beneficial to use SSL for improving classification performance. The cluster assumption of SSL is essentially the notion that a classifier's decision boundary should not pass through high-density regions of the marginal data distribution. This is similar to the large-margin assumption in transductive SVMs and can be used for leveraging unlabeled data. This was done in (Grandvalet and Bengio, 2005) where the classifier is required to produce low-entropy predictions on unlabeled data by adding a loss term which minimizes the entropy of $p(y|x;\theta)$ for unlabeled data.

Arising from the smoothness assumption in SSL – namely that nearby points are most likely of the same class – is the notion of *similarity* and thus many SSL models typically use

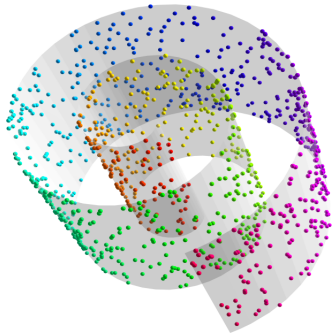


Figure 4.6: Data are assumed to lie on a low-dimensional manifold with nearby points belonging to the same class

some form of *similarity regularization* between points that are assumed to be nearby in feature space. Both labeled and unlabeled data can be used to enforce smoothness and this approach often produces better classification boundaries than using labeled data alone (Chapelle et al., 2006). The notion of similarity is therefore fundamental to many SSL algorithms where the canonical parametric formulation of the SSL loss function takes the following form

$$\sum_{i=1}^{L} l(y_i, f(x_i)) + \lambda \sum_{i,j} \omega_{i,j} g(f(x_i), f(x_j)) \tag{4.28}$$

where $f : \mathcal{X} \to \mathcal{Y}$ is the classifier mapping from input to output space, l is the supervised loss function calculated on the labeled points (which can be a squared loss, hinge-loss or some measure of divergence between predictions and ground truth) and g is the difference in the classifier's output between points weighted by their similarity. The second term represents a *similarity regularizer* incurring a penalty when similar nodes have differing outputs.

Similarity can be captured in a number of ways. In *graph-based* semi-supervised learning, this relationship is encoded via graphs where the nodes represent both labeled and unlabeled points and the weights of the edges reflect the similarity between the nodes (Zhu et al., 2005). The graph then represents the underlying manifold, and a similarity regularizer is applied pair-wise on connected vertices (Figure 4.7). The objective function thus forces labels to be

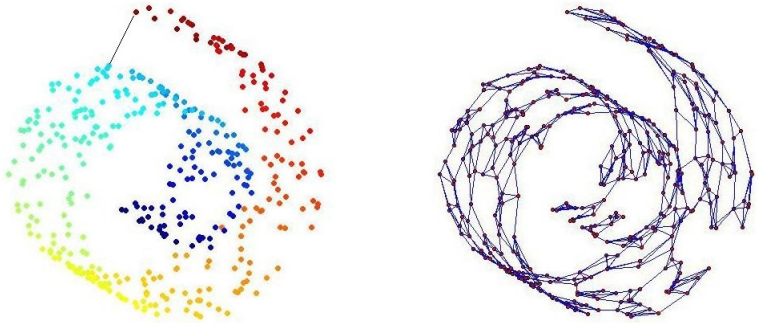


Figure 4.7: A graph can be induced on the low-dimensional manifold. In graph-based SSL, the graph structure is used to enforce the smoothness assumption.

consistent on the manifold. Graph-based SSL algorithms have been successfully applied to tasks such as phone and word classification in automatic speech recognition (ASR) (Malkin et al., 2009; Liu and Kirchhoff, 2013), part-of-speech tagging (Subramanya et al., 2010), statistical machine translation (Alexandrescu and Kirchhoff, 2009), sentiment analysis in social media (Lerman et al., 2009), text categorization (Subramanya and Bilmes, 2008) and many others.

4.7.1 SSL in the Deep Learning Era

The advent of deep learning in recent years has allowed methods that exploit unlabeled data in novel ways, leveraging the representation-learning capacity of DNNs; we give an overview of the various paradigms in the following sections.

Similarity Regularization

Similarity regularization in deep learning is also based on the smoothness assumption, but the notion of similarity can now be expressed in additional ways. For example, a graph can be induced on the input features as well as on the learned representations in the hidden layers; a graph regularizer is then added to the loss function that enforces smoothness on all these graphs. Among the first works to explore this mode of training was (Weston et al., 2008),

where the authors investigated the benefits of addition of graph-based similarity regularizers both over the input features and hidden layers of the network, with noticeable improvements in classification performance over traditional SSL-based methods. A modification to this approach was explored in (Malkin et al., 2009) where a KL-divergence term was used instead of the squared-loss; improved performance was observed on speech data. Other examples of combining graph-based similarity regularization with deep learning include (Yan and Wang, 2009) where a graph induced over the sparse representations that are first learned by a sparse coding is used to enforce similarity regularization. The recently proposed "graph neural machines" in (Bui et al., 2017) showed the efficacy of this method for recurrent neural networks (Hochreiter and Schmidhuber, 1997) as well.

Graph-based semi-supervised deep learning does have some limitations, however. For large-scale learning, the graph has to be sparse, otherwise the graph-construction and computation become too memory intensive to be practical. Additionally to induce a graph, one needs a good measure – or metric – of similarity, and in many applications it is not readily obvious what measure of similarity one must use. Further, the random sampling used in stochastic gradient descent (SGD) – the workhorse of modern deep learning – is not fully compatible with the requirement that a local neighborhood of theagraph be present in the loss function calculation, unless sufficient care is taken to carefully construct the mini-batches needed for SGD. We looked at these issues in detail in a series of works (Thulasidasan et al., 2016; Thulasidasan and Bilmes, 2017) where we showed the efficacy of different heuristic methods that preserve graph structure but also has enough diversity in the mini-batches.

While graph-based deep learning methods were among the first approaches to be successfully applied in the area of semi-supervised deep learning, they have been mostly replaced in recent years by non-graph based methods that exploit other forms of regularization.

Entropy Regularization

A method to force a DNN to output low entropy predictions on unlabeled data was proposed in (Lee, 2013) where the DNN is jointly trained with both labeled and unlabeled data. The

unlabeled data are trained on *pseudo-labels* based on the class with the maximum predicted probability. This is essentially a version of self-training and further it was also shown that this was equivalent to the entropy regularization of (Grandvalet and Bengio, 2005).

It is important here that the predictions on the unlabeled set are phased in gradually through the course of training, since initially the guesses on unlabeled data are not any better than random. The paper achieves this by linearly ramping up the weight on the unlabeled portion. The ramping schedule, initial and final values are hyperparameters that need to be tuned, usually on a per-dataset basis. Empirically, this was shown to achieve good performance on MNIST and CIFAR-10 – scenarios with good class separation – but it is unclear this method works well on more challenging datasets. Further, it also has the same vulnerability as self-training: reinforcement of learning errors.

Consistency Regularization

Modern DNNs that achieve state-of-the-art performance on various classification tasks have hundreds of millions of parameters which make them much more susceptible to overfitting. Indeed, DNNs can fit a completely random assignment of labels to data samples (Zhang et al., 2016), and thus regularization in some form of the other is generally required for performant classification. *Consistency regularization* forces the DNN to make consistent predictions even in the presence of small stochastic perturbations. These perturbations can either be applied to the network itself – by adding random noise to the units or simply disabling a small subset of the weights – or to the data that leave the semantics unchanged, such as rotations, flips and crops for images. The main idea here is that enforcing consistency between predictions on a given data point x and its own perturbed versions does not require knowledge of the true labels, and thus lends itself readily to training in a semi-supervised manner.

Two somewhat concurrently published works (Sajjadi et al., 2016; Laine and Aila, 2016) exploited precisely these ideas for the semi-supervised training of DNNs. In (Sajjadi et al., 2016), the authors consider the problem of semi-supervised learning of image data with convolutional neural networks. Stochastic transformations and perturbations in the form of

random data augmentation, dropout and random max-pooling are applied to the data. The authors propose an unsupervised loss function that takes advantage of the stochastic nature of these methods and minimizes the difference between the predictions of multiple passes of a training sample through the network. Similarity between the original and perturbed versions of the input are enforced both in the input layer, as well as in the hidden layers of the network

More specifically, let N be the number of training samples, over C classes. Assume $\mathbf{f^j}(\mathbf{x_i})$ is the classifier's softmax output over K classesoutputhe i'th training sample during the j'th training pass throuh training sample is perturbed and passed n times through the network. The *transform-stability* loss function for each data sample is:

$$l_{\mathcal{U}}^{\mathrm{TS}} = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \left\| \mathbf{f^j}\left(\mathbf{T^j}\left(\mathrm{x_i}\right)\right) - \mathbf{f^k}\left(\mathbf{T^k}\left(\mathrm{x_i}\right)\right) \right\|_2^2 \tag{4.29}$$

where $\mathbf{T^j}(\mathbf{x_i})$ is a random transformation on the data before the j'th pass through the network. The loss function minimizes the squared output between each possible pair of predictions for a given data point. Note that the above loss function does not prevent the classifier from trivially assigning all unlabeled points to the same class. Thus the authors also propose a *mutual exclusivity* loss that forces each prediction vector to be valid and prevents trivial solutions. This loss function for the training sample x_i is defined as follows:

$$l_{\mathcal{U}}^{\mathrm{ME}} = \sum_{j=1}^{n} \left(- \sum_{k=1}^{C} f_k^j\left(\mathrm{x}_i\right) \prod_{l=1, l\neq k}^{C} \left(1 - f_l^j\left(\mathrm{x}_i\right)\right) \right) \tag{4.30}$$

The authors then use a combined loss function

$$l_{\mathcal{U}} = \lambda_1 l_{\mathcal{U}}^{\mathrm{AE}} + \lambda_2 l_{\mathcal{U}}^{\mathrm{TS}} \tag{4.31}$$

for training and showed that this approach achieves state-of-the-art results on a number of image benchmarks.

In addition to regularizing on transformed versions of data and network, the work in (Laine and Aila, 2016) also proposed to regularize the output of the network against earlier versions of itself – so called "temporal ensembling" or to be more precise, "temporal self-ensembling". The loss functions are similar to that of (Sajjadi et al., 2016) with the squared-loss error being used for regularization. Like the approach in (Lee, 2013), the loss on the unlabeled portion is linearly ramped up because otherwise the network easily gets stuck in a degenerate solution where no meaningful classification of the data is obtained. One of the advantages in this method in terms of running time is that the previous versions of the output predictions are already computed, so these are available for free (in terms of computation; one still has to store these predictions). The authors use an exponentially moving weighed average of these predictions during training, and show state-of-art results on multiple image benchmarks.

Inspired by the consistency regularization on predictions over stochastically perturbed inputs in the above method, the "Mean Teacher" method in (Tarvainen and Valpola, 2017) instead proposes to regularize predictions over an exponential moving average of the model parameters Θ instead of averaging over label predictions. This has the advantage that parameters are continuously averaged every iteration, rather than only once per epoch as is the case when averaging over label predictions, and was shown to outperform the temporal ensembling method described above.

And finally, also very much within the paradigm of consistency regularization, is the "Virtual Adversarial Training" proposed in (Miyato et al., 2018) that uses adversarial training approaches (Szegedy et al., 2013b; Goodfellow et al., 2014b) to perform consistency regularization. A tiny adversarial perturbation r_{adv} is added to the model input x to maximize the change in predicted output. Consistency regularization in this scenario forces the model to output the same prediction as it did on the non-perturbed input. While making the model somewhat robust to adversarial perturbations, it also has the added benefit of being able to leverage unlabeled data.

The methods described above all fall into one of the dominant paradigms of semi-supervised deep learning – regularization based on similarity, entropy minimization or self-consistency.

There are a number of other methods that fall outside these paradigms such as deep generative models (Kingma et al., 2014) that we do not discuss here since we will not be using it when combining abstention with semi-supervised training. Our approach for semi-supervised training will employ a combination of the above paradigms, which we discuss in the following section.

4.7.2 Combining Paradigms for Semi-Supervised Deep Learning

The paradigms for semi-supervised deep learning described above are not mutually exclusive and thus there is no reason one cannot combine them. In fact, this was precisely what was done in the very recently proposed technique called MixMatch (Berthelot et al., 2019) which is currently the state-of-the-art in deep semi-supervised learning on image benchmarks. When trained on CIFAR-10 with only 250 labeled images, MixMatch outperforms the previous state of the art (Virtual Adversarial Training) by a 25% absolute improvement in the error rate (11.08% for MixMatch vs 36.03% for VAT). Note that the fully supervised case on all 50k images has an error rate of 4.13%; this and additional results on other image benchmarks empirically indicate a combination approach can dramatically improve classification performance in semi-supervised learning.

The MixMatch approach combines consistency regularization, entropy minimization and general regularization. During training, stochastic augmentations are first applied to the data, with unlabeled data being augmented multiple times. Predictions on the unlabeled data are then averaged, subsequently sharpened (entropy minimization) and then regularized for consistency (similar to the approaches described earlier). The "Mix" in MixMatch comes from the fact the final step in MixMatch consists of taking convex combinations of inputs and labels using a technique called *Mixup* introduced in (Zhang et al., 2017). The mixing parameter in Mixup is randomly chosen [2], and thus produces a steady stream of new training samples. Training on the resulting augmented and label-smoothed data was shown to consistently

[2]The mixing parameter is sampled from a Beta distribution ensuring that the class probabilities sum to one

improve classifier performance in vision and speech. We shall discuss Mixup in much more detail in Chapter 7 where we investigate its benefits on the predictive uncertainty of deep neural networks.

The different steps in MixMatch produce a batch of $\mathcal{X}$ and $\mathcal{U}$, the mixed labeled and unlabeled data respectively, and introduces a number of additional hyperparameters in this multi-stage process; implementation details, and guidelines on parameter choices are given in (Berthelot et al., 2019). Training this way, the authors report significantly improved classification performance over previous state-of-the-art on various image benchmarks. Further, the ablation studies in the paper show that the greatest single contributing factor to the performance boost is the Mixup component.

4.8 Improving Label-Noise Robustness with the DAC and SSL: A Two-Step Approach

Semi-supervised approaches to label-noise in deep learning has not received much attention. The abstention-based approach for identifying wrongly label data, however, lends itself readily to subsequent semi-supervised learning since what the DAC abstains on are, as we have seen, usually the wrongly labeled samples. This suggests a two-stage approach for learning:

1. Use the DAC – along with PID control if a label noise estimate is available – to first identify mislabeled points.

2. Drop the training labels of the abstained samples from step one, and treat them as unlabeled data, and then proceed with SSL effectively use all the data for learning; this two-stage method is illustrated in Figure 4.8.

A similar two-stage approach is employed in the work of (Ding et al., 2018) which is one of the few works to investigate this technique for label noise. The main difference compared to out proposed method is in the first stage where mislabeled data is identified. The authors try two approaches for this: in the first method, a DNN is pre-trained with the noisy data, and then the most confident predictions of this DNN on the training data are used to identify

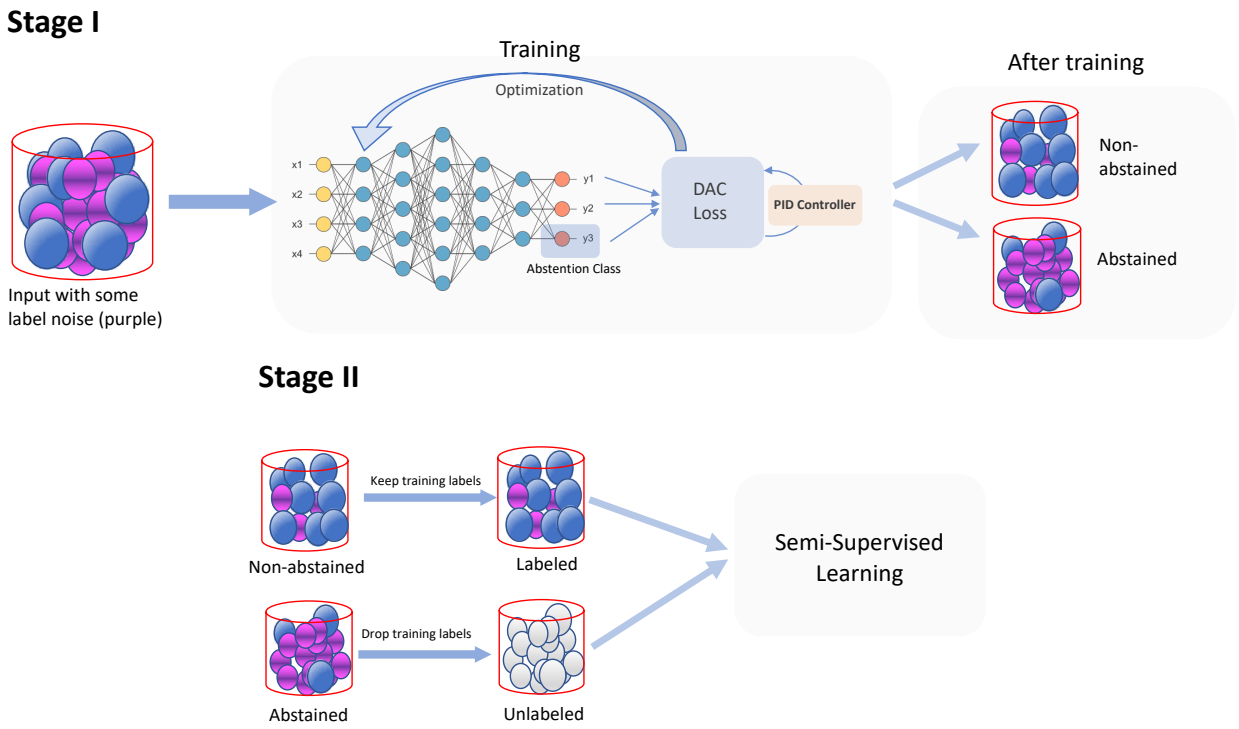


Figure 4.8: Two-stage training with the DAC. In Stage one, the DAC is used to identify noisily labeled training data through abstention. In Stage two, all training data that were abstained on are treated as unlabeled data. The labeled and unlabeled data are then used together for semi-supervised learning.

correctly labeled samples. Confidence is determined by a pre-specified threshold on the winning softmax score. For the other approach, the authors use a portion of the training data that is known to be clean to train a one-vs-rest binary classifier H_k for each class, and then use H_k for class-wise identification of mislabeled points on the rest of the noisy training set. The second stage consists of semi-supervised training, where the labels of the (possibly) mislabeled samples are dropped. The SSL technique of choice in that work was the Temporal Ensembling method (Laine and Aila, 2016) which we discussed earlier.

The problem with using the confidence on the training scores is that, as we have seen, given sufficient training time, DNNs can easily overfit noisy training data with high confidence,

and unless one employs a clean validation set it is generally not possible to determine when the appropriate level of confidence has been reached on the training set. In the second method, a clean, sufficiently large training set has to be manually curated. Further, in a scenario with a moderately large number of classes, training a one-vs-rest binary classifier can lead to a severe class imbalance problem, and sufficient care has to be taken to address this issue. Not surprisingly, the results reported in the paper were all on datasets with a small number of classes – MNIST, CIFAR-10 and Clothing 1M(14 classes). Nevertheless, as this is one of the few works that take an SSL approach to label noise, we will use this as one of our baselines.

4.8.1 Experiments

For the experiments in this section, we use the CIFAR-10 and the more challenging CIFAR-100 datasets, and as before, randomize an increasing fraction of labels. Since MixMatch needs significantly more training time than a regular, fully supervised DNN, for these experiments we limit ourselves to these two datasets (50k training samples). We use a PID-controlled DAC to stabilize to an abstention set-point, and in order to demonstrate the effectiveness of sampling for estimating label noise, we set the abstention set-point to the noise rate that was estimated by examining only 400 training samples.

The first-stage DAC-PID is trained on two different architectures (ResNet-34 and WideResNet-28x10) that were used in the experiments in Section 3.4. For the semi-supervised learning with MixMatch in the second stage, we use the same architecture and best performing hyperparameters that were reported in the MixMatch paper (Berthelot et al., 2019).

Results are shown in Table 4.1. The baseline is the case where we train on the noisy data, while DAC refers to the two-step method described in Chapter 3 where we train with regular cross-entropy after first using the DAC to identify mislabeled points; the numbers are reproduced from the experiments in Section 3.4. DAC-SSL is the two-stage semi-supervised method propsoed here. As we can see, using all the training data in a semi-supervised manner after identifying and dropping the labels of the potentially mislabeled data leads to significant performance gains, especially in scenarios where the label noise is high. Our method also

Dataset	Method	Label Noise Fraction			
		0.2	0.4	0.6	0.8
CIFAR-10 (ResNet-34)	Baseline	88.94	85.35	79.74	67.17
	DAC	92.91	90.71	86.30	74.84
	Ding et al.	84.5	-	75.8	
	DAC-SSL	**95.14**	**94.08**	**90.91**	**84.86**
CIFAR-10 (Wide Res-Net 28x10)	Baseline	91.53	88.98	82.69	64.09
	DAC	93.35	90.93	87.58	70.8
	DAC-SSL	**95.24**	**93.69**	**90.94**	**79.25**
CIFAR-100 (ResNet-34)	Baseline	69.15	62.94	55.39	29.5
	DAC	73.55	66.92	57.17	32.16
	DAC-SSL	**76.49**	**71.47**	**64.22**	**43.6**
CIFAR-100 (Wide Res-Net 28x10)	Baseline	71.24	65.24	57.56	30.43
	DAC	75.75	68.2	59.44	34.06
	DAC-SSL	**76.81**	**71.09**	**63.89**	**48.89**

Table 4.1: Results for abstention training followed by semi-supervised training using the MixMatch algorithm. The DAC is trained on different datasets and architectures. Downstream SSL training is always done on the WideResNet 28x10 architecture. DAC and Baseline results reproduced from Chapter 3

outperforms the results reported in (Ding et al., 2018) by a large margin, empirically proving the efficacy of an abstention-based approach followed by semi-supervised learning. We leave experiments on larger, real word noisy datasets for future work.

4.9 Conclusion

In this chapter we showed how the DAC can be extended in multiple ways. Using the theoretical results established in Chapter 3, we were able to successfully apply a PID-controller approach to stabilize abstention behavior. We further discussed how classical statistics can help us determine the label-noise rate in a dataset and enable us to set an appropriate abstention set-point, which makes this method practical in real-world scenarios. And finally we demonstrated how abstention can be easily combined with semi-supervised deep learning to significantly improve training robustness and classifier performance in the presence of label noise.

Part II

IMPROVING PREDICTIVE UNCERTAINTY IN DEEP LEARNING

Chapter 5

UNCERTAINTY IN DEEP LEARNING: AN OVERVIEW

In this part of the thesis, we study the problem of predictive uncertainty of deep neural networks. We begin by exploring the various issues in modeling uncertainty in deep learning – particularly, that of overconfidence and miscalibration – and discuss existing approaches, including methods for classification with abstention (or rejection), Bayesian approaches to deep learning and other recent work in the field.

5.1 Introduction

Owing to their dramatic successes in perceptual tasks like vision and speech recognition, DNNs are also being increasingly deployed in domains like medical diagnosis (Miotto et al., 2016; Litjens et al., 2017) and autonomous driving(Falcini et al., 2017; Huval et al., 2015) where it is not only important to make accurate predictions, but also critical to have a *reliable measure of confidence* in the model's prediction. An erroneous prediction that should have otherwise been flagged for human intervention – because the system has not robustly learned when it is likely to get the wrong answer – can have severe consequences; the major challenges in deep learning, going forward, are thus going to be in issues of uncertainty and trust-worthiness of such classifiers.

Before discussing existing approaches to quantifying uncertainty, it is instructive to briefly discuss the various sources of uncertainty in machine learning. In Part I, we saw how label noise in the training data can lead to increased predictive errors; the abstention formulation we developed allowed us to identify features that correlate with unreliable labels (Section 3.3), and thus gave us a mechanism to abstain from prediction when encountering such features during test time. However, even a model trained on data that is free of labeling

errors can generally *never be completely certain* in its predictions. This is especially the case when the model is presented with an object that has features common to more than one class (such as wolf and husky dog), or one that lies away from the distribution of training data (out-of-distribution or unknown categories); in such cases, one expects the classifier to output low confidence predictions.

Further, there is also inherent uncertainty in the model parameters itself. Multiple models (neural networks in our case) might exist that explain the data (which one should we trust?), but the choice of the structure of the model is often decided in advance and somewhat arbitrary. Because deep models are usually trained using the principle of Maximum Likelihood Estimation (MLE), we generally end up with only *point estimates* of parameters – i.e., a *single* model – without any notion of uncertainty on the model weights themselves. Bayesian approaches (which we discuss in Section 5.5 of this Chapter) attempt to tackle this directly by computing the distribution on model parameters; however, the sheer size of a modern deep neural network – which often have hundreds of millions of parameters – usually make such approaches infeasible.

The focus of our work in uncertainty quantification in this thesis is on *predictive uncertainty* – as opposed to model uncertainty – and our approach will be a non-Bayesian one. We will exploit the fact that the output of a DNN – which can be mathematically interpreted as probabilities – can also be treated as actual probabilities of a correct prediction, but only if the output scores for a classification decision match the actual likelihood of a correct prediction; in other words, the DNN has to be *well calibrated*. But the capacity of modern DNNs makes them easily prone to overconfident – and thus, miscalibrated –predictions. Hence a significant part of our discussion in this chapter and the next will be devoted to discussing and improving calibration in deep learning; we begin, however, with a look at existing approaches to classification with rejection.

5.2 *Classification with Rejection*

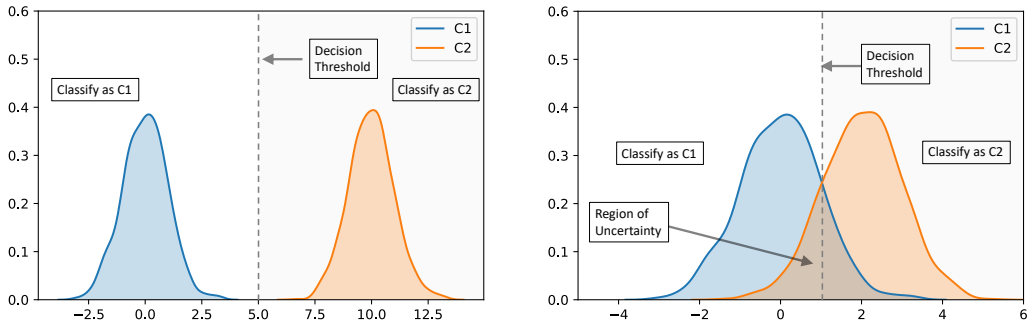
Since learning systems have been around for multiple decades, there have been extensive theoretical and empirical investigations into rejection (or abstention)-based classification. Before discussing the work in this field, we take a look at the basic idea in these systems. Such classifiers can be more easily understood in the context of binary classification: let x be an observation that can only belong to one of two possible classes, $C1$ or $C2$, with class priors of π_1 and π_2, and probability density over features $p_1(x)$ and $p_2(x)$ respectively. If all these statistics – class priors and feature densities – are known, then the optimal decision rule, using Baye's rule – is given by:

$$y_i(x) = \frac{\pi_i p_i(x)}{\pi_1 p_1(x) + \pi_2 p_2(x)}, \quad i = 1, 2 \tag{5.1}$$

and

$$c(x) = \max\{y_1(x), y_2(x)\} \tag{5.2}$$

where $y_i(x)$ denotes the posterior probability of class C_i and $c(x)$ is the decision. The risk of a false classification is given by $r(x) = 1 - c(x)$ and is called the "Bayes Optimal Risk". The Bayes rule thus always assigns x to the class for which the computed likelihood is higher. An



(a) (b)

Figure 5.1: An illustration of likelihood-based classification

illustration of likelihood-based classification is given in Figure 5.1. When class distributions $p_1(x)$ and $p_2(x)$ are non-overlapping, the optimal decision rule always picks the right class (Figure 5.1a). However, most real life scenarios are more akin to the situation in Figure 5.1b where there is always some degree of uncertainty in the predicted class. This can be due to the fact one almost never has access to the actual distributions over features or the true class priors (which have to be determined empirically). Even with perfect knowledge of feature densities and priors, there might still be uncertainty if the features do not have sufficient discriminatory power for a perfect classification (for example, predicting gender just given height in a population of adults). Such situations inevitably gives rise to detection error and thus uncertainty regarding the decision (shown by the shaded region in Figure 5.1b). In these cases, an appropriate detection threshold (shown by the dotted vertical line) based on an acceptable application-specific risk has to be manually set by the user. When the probability of the predicted class falls below this threshold, the prediction is rejected. Note that when the true distributions are not perfectly separated, one inevitably ends up rejecting actual instances of the class in return for a lower risk of false prediction; the aim then is to maximize coverage (or minimize rejection rate), while still getting high predictive accuracy.

The above ideas form the basis of most approaches to rejection-based classification in machine learning, some of which date back to over 60 years. One of the first works in this area is (Chow, 1957), which describes an optimal rejection classifier for Optical Character Recognition (OCR) systems based on knowing the *a priori* probabilities of characters; these ideas were further extended in (Chow, 1970) that describes an optimal error-reject trade-off. Rejection systems in the context of K-nearest-neighbor classifiers were considered in (Hellman, 1970) based on a combination of Baye's decision rule and voting among the K nearest neighbors, the threshold here being the number of pre-specified neighbors $k' \leq K$ that have to agree for an acceptance decision; thresholds can also be applied on the distance d from the nearest neighbor.

In the context of Support Vector Machines (SVMs), thresholds have been applied on the output $|f(x)|$ of the SVM, since the output is proportional to the distance of x from

the optimal separating hyperplane (OSH); in this case, points are rejected if they are *too close* to the OSH. Such mechanisms formed the basis of the work described in (Bartlett and Wegkamp, 2008) and (Grandvalet et al., 2009), with the former also taking into account the cost of rejection. An abstention model based on abstention budget and misclassification cost was also described in (Pietraszek, 2005). The work in (Bartlett and Wegkamp, 2008) was generalized in (Cortes et al., 2016) which studied rejection mechanisms on both kernel-based hypothesis classes as well as confidence-based rejection, with generalization guarantees given in terms of Rademacher complexity (Bartlett and Mendelson, 2002) of the classifier.

The above work was in the context of shallow learning models. In the case of neural networks, early work by (Cordella et al., 1995) explored the classification reliability of multilayer perceptrons (MLP) which considered the acceptable risk of misclassification versus the added burden of rejection including cases where the misclassification cost is high; optimal reject threshold values are then derived. This work was extended in (De Stefano et al., 2000) which analyzed the distribution of scores on correct and incorrect predictions of a trained MLP model and thereby determined an optimal rejection threshold.

In the case of deep learning, a selective classification method was described in (Geifman and El-Yaniv, 2017) which introduced the concept of selection with guaranteed risk (SGR). Here, the predictions of a pre-trained model f are filtered through a *selection function* $g : \mathcal{X} \to \{0, 1\}$ so as to abstain on predictions that are likely to be wrong while maximizing coverage as follows:

$$(f, g)(x) \triangleq \begin{cases} f(x), & \text{if } g(x) = 1 \\ \text{don't know}, & \text{if } g(x) = 0 \end{cases} \tag{5.3}$$

Given a user-specified coverage denoted by $\phi(f, g) \triangleq E_P[g(x)]$ – where P is the distribution over $\mathcal{X}, \mathcal{Y}$ – selective risk is defined as:

$$R(f, g) \triangleq \frac{E_P[\ell(f(x), y)g(x)]}{\phi(f, g)} \tag{5.4}$$

The performance of the classifier can then be characterized by the trade-off between risk

and coverage. The selection function g is constructed such that the risk is bounded as follows:

$$\Pr_{S_m}\{R(f,g) > r^*\} < \delta \tag{5.5}$$

Given a notion of confidence κ_f induced by f, g can then be parametrized by a confidence threshold θ as follows:

$$g_\theta(x) = g_\theta\left(x|\kappa_f\right) \triangleq \begin{cases} 1, & \text{if } \kappa_f(x) \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

Confidence scores on which the model is tuned are obtained either by the winning softmax outputs or by averaging the scores of multiple inference passes with dropout turned on at test time (Gal and Ghahramani, 2016).

Note that all of the above assume models that have already been trained, and apply confidence thresholds after training; in other words the model *does not learn to reject* as part of its training. Embedded options for rejection integrated in the training phase were explored for SVMs in (Fumera and Roli, 2002) where multiple parallel hyperplanes are learned to determine high and low-confidence regions. More recently, a DNN that learns an optimal rejection function as part of training was proposed in (Geifman and El-Yaniv, 2019), which is an extention of the authors' work in (Geifman and El-Yaniv, 2017) covered above. This deserves special mention as it is somewhat conceptually similar to our abstention framework introduced in Chapter 3. In this work the classifier and the selector function are jointly learnt using a 3-headed architecture called SelectiveNet: the classification head (f) for non-abstained samples, the selection (or abstention) head (g), and an auxiliary head (h) that is oblivious to abstention and makes predictions on all samples. Given a labeled set $\mathcal{D}$, the combined loss function is defined as:

$$\mathcal{L} = \alpha\mathcal{L}_{(f,g)} + (1-\alpha)\mathcal{L}_h \tag{5.7}$$

where

$$\begin{aligned}\mathcal{L}_{(f,g)} &\triangleq \hat{r}_\ell\left(f, g|\mathcal{D}\right) + \lambda\Psi\left(c - \hat{\phi}\left(g|\mathcal{D}\right)\right) \\ \Psi(a) &\triangleq \max(0, a)^2\end{aligned} \tag{5.8}$$

and

$$\mathcal{L}_h = \hat{r}\left(h|\mathcal{D}\right) = \frac{1}{m}\sum_{i=1}^{m}\ell\left(h\left(x_i\right), y_i\right) \tag{5.9}$$

Here c is the user-specified coverage and $\hat{\phi}\left(g|\mathcal{D}\right)$ is the empirical coverage calculated as

$$\hat{\phi}\left(g|\mathcal{D}\right) \triangleq \frac{1}{m}\sum_{i=1}^{m} g\left(x_i\right) \tag{5.10}$$

where the selector $g(x)$ is a single neuron with a sigmoidal activation. The authors note that the auxiliary head h is needed to enable useful learning, otherwise the model abstains on a random fraction of the training set. At test time SelectiveNet predicts $f(x)$ if and only if $g(x) \geq 0.5$, and abstains otherwise.

One of the implicit assumptions in all of the above is that a higher output score is more likely to be indicative of a correct prediction. The learning capacity of DNNs, however, present challenges to methods that rely on the output of a DNN as a proxy for confidence. DNNs, it turns out, are not only overly confident on predictions on in-distribution data, but also tend to become *pathologically overconfident* on samples that lie far away from the training set; we examine these issues in the next section.

5.3 Overconfidence and Miscalibration in Deep Learning

Consider again the output of a K-class DNN, where the penultimate layer's output $h(.)$ is converted to a probability distribution over the K classes by passing it through a *softmax* layer

$$s_i(z) = \frac{\exp(h_i(z))}{\sum_{k=1}^{K}\exp(h_k(z))} \tag{5.11}$$

This can be viewed as the DNN's estimate of the class conditional probabilities $p(y|x)$,

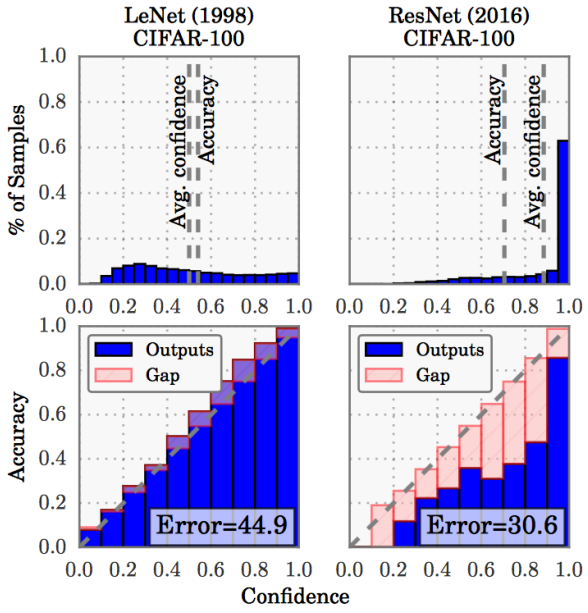


Figure 5.2: Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Figure reproduced from (Guo et al., 2017)

that is, $f_i(\Theta, x) \in \triangle^{K-1}$, the $K-1$-dimensional probability simplex. While mathematically, these indeed satisfy the requirements of a probability distribution over K classes, the output can only be treated as an actual probability if the DNN is *well calibrated*, in which case the softmax output is indeed the actual likelihood of correctness.

However, recent work (Guo et al., 2017) provide significant empirical evidence that modern deep neural networks are poorly calibrated, with depth, weight decay and batch normalization all influencing calibration. For example, the histogram plots in Figure 5.2 illustrates this phenomena: older neural networks like LeNet (LeCun et al., 1998), even though less accurate than modern architectures, tend to be better calibrated. Modern architectures, however, are prone to overconfidence, meaning accuracy is likely to be lower than what is indicated by the predictive score.

This phenomenon of overconfidence has been observed in a wide variety of deep archi-

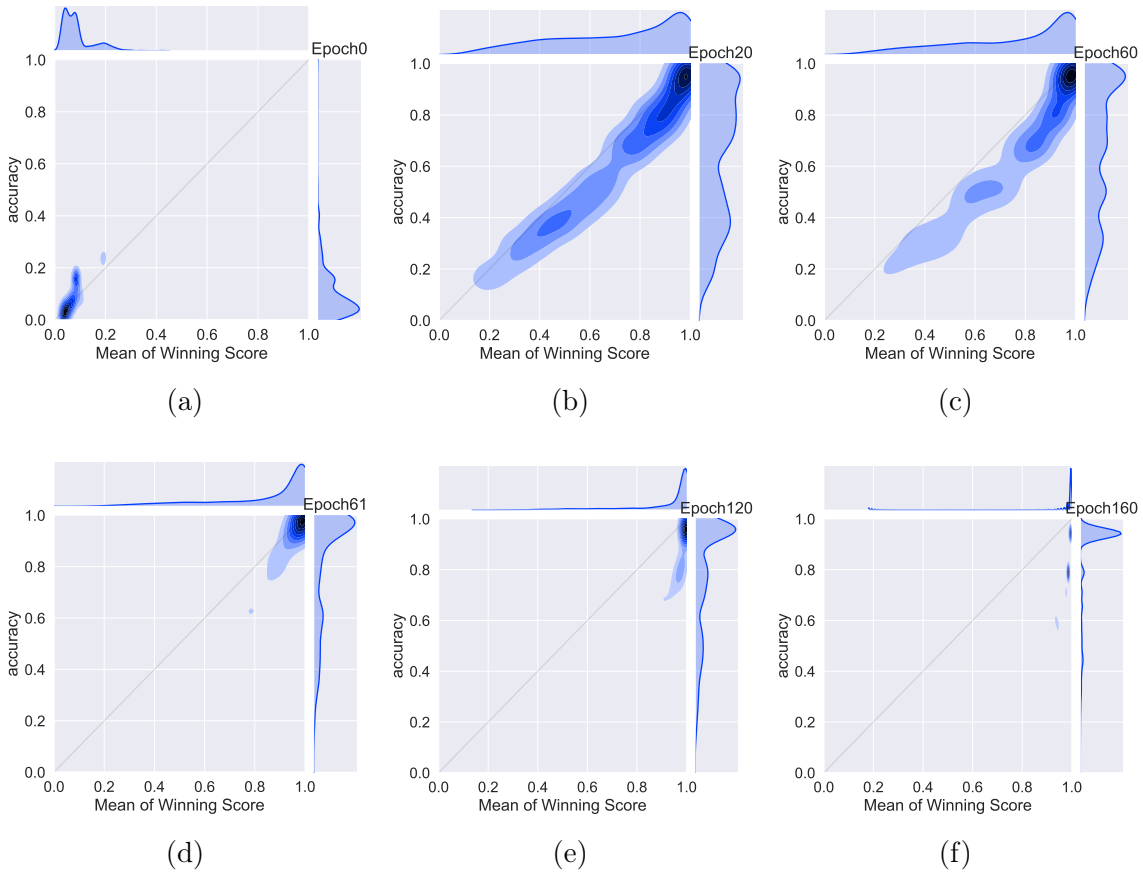
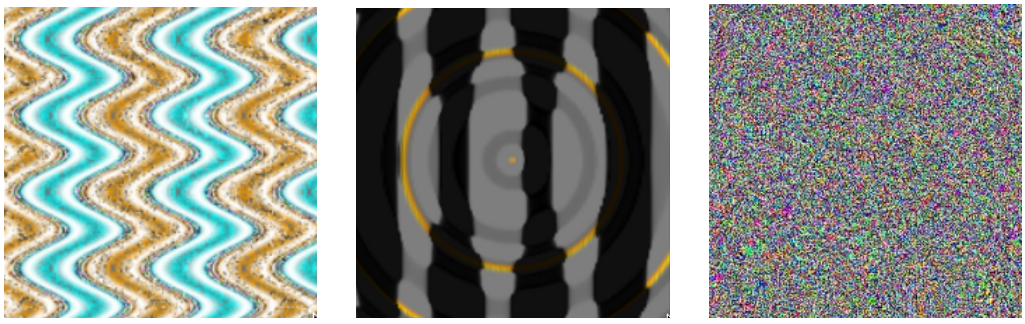


Figure 5.3: Accuracy vs confidence (captured by mean of the winning score) on the CIFAR-100 validation set at different training epochs for the VGG-16 deep neural network. The DNN moves from underconfidence, at the beginning of training, to overconfidence at the end. A well-calibrated classifier would have most of the density lying on the $x = y$ gray line

tectures. Figure 6.1 shows a series of joint density plots of the average winning score and accuracy of a VGG-16 network over the CIFAR-100 validation set, plotted at different epochs. Both the confidence (captured by the winning score) as well as accuracy start out low and gradually increase as the network learns. However, what is interesting – and concerning – is that the confidence always leads accuracy in the later stages of training. Towards the end of training, accuracy saturates while confidence continues to improve resulting in a very sharply peaked distribution of winning scores.

To mitigate the above problem, (Guo et al., 2017) propose a post-training calibration method – called *temperature scaling* – to produce output scores that have a better predictive value of the actual answer. However, as we shall see in later sections, while temperature scaling does produce well calibrated classifiers on in-distribution data, it is unable to accurately discriminate between instances of in and out-of-distribution data.

In fact, lack of robustness to distributional shift is a significant concern in machine learning, and particularly so in deep learning owing to the increasing usage of such models in safety critical applications; for example the phenomena of overconfident DNN predictions was observed for medical diagnosis tasks (Leibig et al., 2017). Work described in (Nguyen et al., 2015) shows how DNNs can be easily fooled (in a non-adversarial manner) into making high confidence predictions even on objects that lack *any* visual resemblance to the predicted class; we show two such examples in Figures 5.4a and 5.4b. Additionally, Figure 5.4c shows a high confidence prediction of "cat" on a purely random noise image on a VGG-16 network trained on the STL-10 dataset.



(a) (b) (c)

Figure 5.4: A sample of images where a DNN makes very high confident predictions. In this case, the predictions were "starfish" (a),"king penguin" (b) and "cat"(c), all with softmax confidence > 0.99). Images (a) and (b) reproduced from `http://www.evolvingai.org/fooling`

While there exists a rich body of literature on adversarial perturbations((Papernot, 2018)) – which are specifically designed to produce high confident erroneous predictions – these results suggest that one need not even go the adversarial route to induce pathological

behavior. Clearly, threshold-based techniques will fail to work here since the output prediction confidence shows near certainty; it is worthwhile then to explore why such behavior happens in the first place.

Most neural networks use some form of non-linear activation function in the hidden layers. While early networks used the sigmoid function, most modern DNNs use the *Rectified Linear Unit* (or ReLU (Zeiler et al., 2013)) which has shown to produce better classification performance, being more resistant to the gradient-vanishing problem (Pascanu et al., 2013) in very deep networks; indeed the ReLU has become fairly ubiquitous in modern architectures.

Recent work (Hein et al., 2018) has however also suggested that the ReLU (and its variants) might be contributing to the overconfidence problem. Using the fact that ReLU networks produce piecewise affine classifiers (Arora et al., 2016), the work in (Hein et al., 2018) argues that essentially any neural network which results in a piecewise affine classifier function produces arbitrarily high confidence predictions far away from the training data, implying that techniques that employ confidence thresholding are likely to fail. However, their analysis assumed an unbounded input domain, and is not directly applicable to bounded domains like images where the pixel values are normalized to be in $[0, 1]$. But in addition to the ReLU aspect, the phenomena of high confidence predictions are also an outcome of the final softmax layer, as pointed out in the aforementioned work as well as in other places (Subramanya et al., 2017); we turn our attention to this aspect in the next section.

5.4 A Closer Look at Softmax

The softmax output function, concisely defined as

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{5.12}$$

is, as discussed previously, used as the final layer in most classification DNNs to convert the real, continuous activation values in a neural network to a discrete probability mass distribution over K classes. While the use of the exponential function works well when

training using maximum likelihood estimation (since the negative log-likelihood formulation of MLE cancels out the exponentiation and results in a convenient analytical expression for gradients), the softmax is also considered to be prone to inflating the probability of the predicted class resulting in overconfident (low entropy) predictions and hence, unreliable uncertainty estimates. In fact, as we saw in Figure 5.4 deep neural networks can predict confidently even when presented with random noise or fooling images.

But even simply scaling the input can increase confidence as follows: let $\mathbf{z} = [z_1, z_2, ...z_k]$ be the input into the softmax unit, with z_c being the pre-activation of the winning class. Consider a *scaled* version of z, namely αz, where $\alpha > 1$.

$$p(y = c|\alpha z) = \frac{\exp \alpha z_c}{\sum_k \exp \alpha z_k} \tag{5.13}$$

$$= \frac{1}{1 + \sum_{k,k \neq c} \exp \alpha (z_k - z_c)} \tag{5.14}$$

$$\rightarrow 1 \text{ as } \alpha \rightarrow \infty \text{ (since } z_k < z_c) \tag{5.15}$$

Thus the softmax probability can be brought arbitrarily close to 1 even in a shallow network by simply scaling the input. For example, by increasing the brightness of an image, one can end up with increased softmax scores of the winning class even though the features and semantics of the image are unchanged; in the next section, we briefly look at an alternative to the softmax layer that avoids such pathologies.

5.4.1 An Alternative to Softmax: The Ratio Semi-Definite Classifier

An alternative to the softmax layer designed to temper overconfidence and encourage low entropy posteriors when the network is presented with data away from the region of the training data was proposed in (Malkin and Bilmes, 2008). Output probabilities were expressed as a *ratio of semi-definite polynomials* which avoids the fast growing nature of the exponential in the softmax. This work was inspired by the semi-definite linear probabilistic models discussed in (Crammer and Globerson, 2012), but extended to a non-linear settings

in (Malkin and Bilmes, 2008) as well as with a multilayer perceptron (Malkin and Bilmes, 2009).

Concretely, the ratio semi-definite classifier (RSC) uses the following probabilistic model (Malkin and Bilmes, 2008).

$$p(y|x) = \frac{(x - d_y)^T A_y (x - d_y)}{\sum_k (x - d_k)^T A_k (x - d_k)} \tag{5.16}$$

where the x's are inputs into the RSC layer. A_k is a positive semi-definite matrix ($\forall k$), associated with class k. In order to be a valid probability distribution, (Malkin and Bilmes, 2008) forces the condition $A_k \succeq 0$ by parameterizing A_k as $A_k = B_k B_k^T$. d_k's are *shift vectors* associated with class k; (Malkin and Bilmes, 2008) referred to these as *anti-means*, since in general d_k will be pushed away from the mean of class k. Then the parameters to be learned during training are the class-specific B's and d's; the model is then trained with the objective function that includes the standard cross-entropy term as well as regularizers for the shift vectors.

Let us assume that the class-specific shift vectors are all $\mathbf{0}$ and for simplicity, assume x has been normalized, i.e., $||x|| = 1$. Since $A_k \succcurlyeq 0$, we have $\lambda_{max}(A_k) \geq x^T A_k x \geq \lambda_{min}(A_k) \geq 0$ where $\lambda_{max}(A_k)$ and $\lambda_{min}(A_k)$ are respectively the largest and smallest eigenvalues of A_k

Let c be the winning class, i.e $c = \text{argmax}_k \, x^T A_k x$ Then,

$$p(y = c|x) = \frac{x^T A_c x}{\sum_k x^T A_k x} \tag{5.17}$$

$$= \frac{1}{1 + \sum_{k, c \notin k} x^T A_k x} \tag{5.18}$$

$$\leq \frac{1}{1 + \sum_{k, k \neq c} \lambda_{min}(A_k)} \tag{5.19}$$

If A_k's are full rank, then $p(y|x)$ is always bounded by $p < 1$, since $\lambda_{min}(A_k) > 0$, $\forall k$. Also, scaling input by α makes no difference since α cancels out, i.e., $p(y = c|z) = p(y = c|\alpha z)$ where z is the input into the RSC unit. Thus, unlike the softmax, the RSC is bounded (in

the full-rank case) which suggests that the RSC might be immune to the overconfidence in softmax networks. The softmax itself, however, can be bounded by label-smoothing where an ϵ probability mass is always present in the labels of the classes other than the ground-truth class; we explore the effects on predictive uncertainty of such techniques in much more detail in Chapter 6. Nevertheless, the bounded nature of the RSC, makes this a worthwhile line of exploration and leave it for future work; we end this chapter by briefly discussing *Bayesian* approaches to uncertainty modeling.

5.5 Bayesian Deep Learning

Traditionally, uncertainty quantification has been the domain of Bayesian methods. In the Bayesian approach, what is of interest is *model uncertainty* – since multiple models might be able to explain the data – from which predictive uncertainty can then be obtained. For example, Gaussian processes (Rasmussen and Nickisch, 2010) define probability distributions over functions, which then give principled estimates of which models are likely to generalize well from the training data. The work in (Neal, 1995) showed that by placing a probability distribution over each weight, a Gaussian process can be recovered in the limit of infinitely many weights. For a finite number of weights, model uncertainty can still be obtained by placing a distribution over each weight (Gal, 2016). This approach, termed Bayesian Neural Networks, or more recently, Bayesian Deep Learning, however has not seen wide adoption due to the large number of parameters that make up a modern DNN; indeed, full Bayesian estimation has been generally infeasible due to the difficulty of density estimation in very high dimensional spaces.

However, recent work has exploited the stochastic nature of training deep models to perform *approximate Bayesian inference*. For example, in (Gal and Ghahramani, 2016), the authors show how using dropout (Srivastava et al., 2014) during the inference stage of a pre-trained classifier is mathematically equivalent to approximate Bayesian inference in deep Gaussian processes. Since dropout is relatively simple to implement, and does not increase the computational time in any significant way, ensemble models using Monte Carlo dropout –

where predictions are averaged over multiple dropout passes– are being adopted in real world applications (Leibig et al., 2017).

The work in (Osawa et al., 2019) proposes a practical method to train deep models with natural-gradient (Amari, 1998) variational inference. By applying techniques such as batch normalization, data augmentation and distributed training, the authors show a scalable approach to Bayesian learning, although this still requires more computational resources that a straightforward MLE estimation.

Also, recently, in (Maddox et al., 2019) the authors exploit SGD iterates for Bayesian learning by treating these iterates as samples from some distribution. A Gaussian distribution over the weights derived from these iterates is used to form an approximate posterior distribution over the DNNs. During test-time, weights are then then sampled to perform Bayesian model averaging and uncertainty estimates. While more practical than full-fledged density modeling estimates like Markov-Chain Monte Carlo (Wang and Yeung, 2016), this is still significantly more computationally expensive than a non-Bayesian approach.

It turns out many aspects of training deep models also have a Bayesian interpretation. For example, the method described above in (Maddox et al., 2019) exploits the fact that SGD itself can be viewed as approximate Bayesian inference, a result proved in(Mandt et al., 2017). Also, standard MLE training itself along with L_2 norm regularization of the weights is equivalent to Maximum A-Posteriori estimation when applying a Gaussian prior on the weights (Goodfellow et al., 2016); note that this still only gives a point estimate (since MAP estimation gives the *mode* of the posterior distribution) and is not a full-fledged Bayesian approach.

Combining Bayesian-inspired methods with adversarial training, the authors in (Lakshminarayanan et al., 2017) show how an ensemble of classifiers can show improved performance over pure dropout based approaches and strongly argue in favor of a multi-classifier approach to uncertainty estimation. While the techniques were not directly developed from a Bayesian standpoint, it has been argued that deep ensembles can be seen as approximate Bayesian marginalization (Wilson, 2020).

Yet other Bayesian-inspired methods do not attempt to estimate model density over the entire parameter space of the network, but limit themselves to the final few layers. For example the approach described in (Subramanya et al., 2017) aims to estimate a class conditional density of the pre-softmax activations $P(z|y_i)$ on the validation set; one can then apply Bayes rule to determine $P(y_i|z)$.

In (Jiang et al., 2018a) attempts to perform density modeling on the input by filtering out $1-\alpha$ fraction of points that lie in low density neighborhoods based on k-nearest neighbors. A *trust* score is then defined which measures the agreement between the classifier and a modified nearest neighbor classifier defined on the α-high-density set. In (Sensoy et al., 2018) the softmax point estimates output by the DNN are converted to a *distribution* over the point estimates by applying Dirichlet priors, the parameters of which are dependent on the network.

While the above methods have made Bayesian approaches more practical, it isn't clear whether these offer a clear advantage over standard training methods that already incorporate significant stochasticity in the learning process(Mitros and Mac Namee, 2019). Note that deep learning is a "big data" approach; this allows us to exploit the law of large numbers via concentration inequalities (Boucheron et al., 2003) that can already provide strong uncertainty bounds in many settings.

In summary, while Bayesian approaches offer a principled way to compute uncertainty, computational concerns and real-world performance so far have overshadowed their theoretical advantages. The methods we describe in the following chapters adopt a non-Bayesian approach, and our focus will be on *predictive uncertainty* – as opposed to model uncertainty. We will exploit the fact that the output of a DNN – which can be mathematically interpreted as probabilities – can also be treated as actual probabilities of a correct prediction. However, as discussed before, such an interpretation is only justified if the DNN is well-calibrated. In the next chapter, we take a closer look at miscalibration and show how to improve predictive uncertainty in a computationally feasible manner.

Chapter 6

MAKING THRESHOLD-BASED ABSTENTION WORK: IMPROVING CALIBRATION IN DEEP NEURAL NETWORKS

In this chapter, we take a closer look at one of the causes of overconfident predictions in deep neural networks: the practice of training with *hard labels*, where all the probability mass in the label is in one class. We discover that label smoothing techniques — in particular, a recently proposed data augmentation technique called mixup (Zhang et al., 2017) — lead to better calibrated models. Label smoothing has so far been studied in the context of improving model accuracy; ours is the first to show its benefit on calibration. Once the models are better calibrated – meaning, output scores are actually indicative of accuracy – threshold-based abstention can be applied for improved predictive risk both on in and out-of-distribution data. This is joint work with Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya and Sarah Michalak and appeared in the proceedings of NeurIPS 2019 (Thulasidasan et al., 2019b).

6.1 Are DNNs Trained to be Overconfident?

In the previous chapter we saw how DNNs tend to produce overconfident predictions, both on in and out-of-distribution data. The work in (Guo et al., 2017) provided significant empirical evidence that modern DNNs are poorly calibrated with depth, weight decay and batch normalization all influencing calibration. Recall the joint density plots of confidence and accuracy at various epochs on a VGG-16 network on the CIFAR-10 dataset that we depicted in Section 5.3, reproduced here in the top row in Figure 6.1. Both the confidence (captured by the winning score) as well as accuracy start out low and gradually increase as the network learns. However, what is interesting – and concerning – is that the confidence always leads accuracy in the later stages of training. Towards the end of training, accuracy

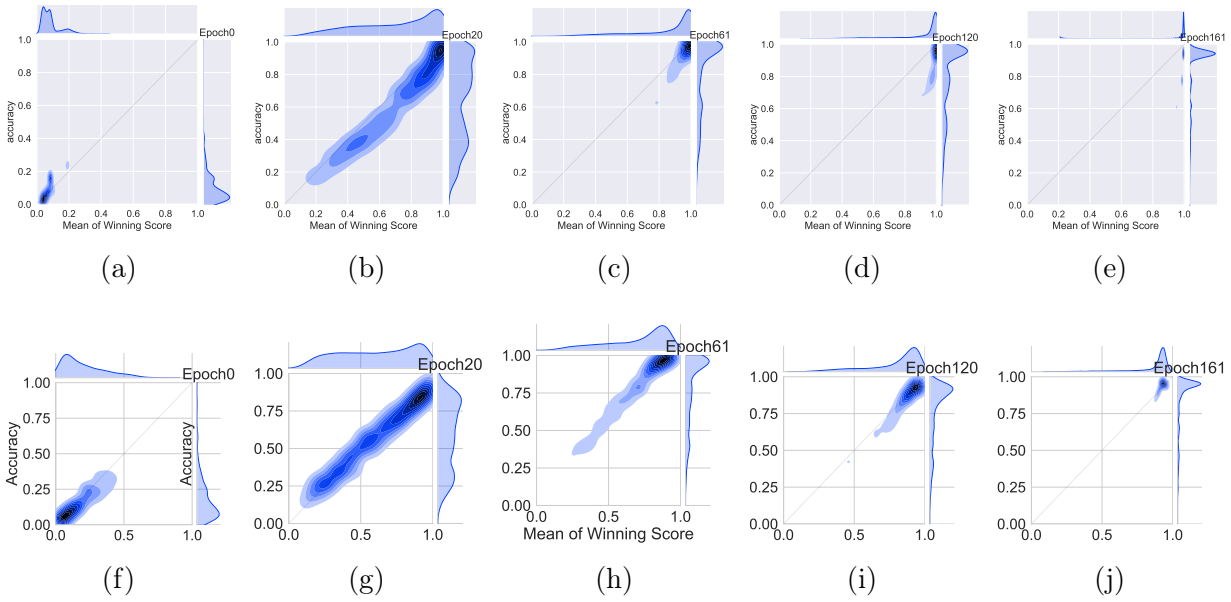


Figure 6.1: Joint density plots of accuracy vs confidence (captured by the mean of the winning softmax score) on the CIFAR-100 validation set at different training epochs for the VGG-16 deep neural network. **Top Row:** In regular training, the DNN moves from under-confidence, at the beginning of training, to overconfidence at the end. A well-calibrated classifier would have most of the density lying on the $x = y$ gray line. **Bottom Row:** Training with mixup on the same architecture and dataset. At corresponding epochs, the network is much better calibrated.

saturates while confidence continues to increase resulting in a very sharply peaked distribution of winning scores and an overconfident model.

Most modern DNNs, when trained for classification in a supervised learning setting, are trained using one-hot encoded labels that have all the probability mass in one class; the training labels are thus zero-entropy signals that admit no uncertainty about the input. The DNN is thus, in some sense, *trained to become overconfident*. Hence a worthwhile line of exploration is whether principled approaches to label smoothing can somehow temper overconfidence. The effects of label smoothing and related work has been explored before (Szegedy et al., 2016; Pereyra et al., 2017) in the context of improving classification accuracy; in this work, we carry explore their effects on calibration, focussing on the recently proposed *mixup* (Zhang

et al., 2017) method of training deep neural networks. In mixup, additional synthetic samples are generated during training by convexly combining random pairs of images and, importantly, their labels as well. While simple to implement, it has shown to be a surprisingly effective method of data augmentation: DNNs trained with mixup show noticeable gains in classification performance on a number of image classification benchmarks. However neither the original work nor any subsequent extensions to mixup (Verma et al., 2018; Guo et al., 2018; Liang et al., 2018a) have explored the effect of mixup on predictive uncertainty and DNN calibration; this is precisely what we address in this paper.

Our findings are as follows: DNNs traind with mixup – and even standard label smoothing described in (Szegedy et al., 2016) – are significantly better calibrated, meaning the predicted softmax scores are much better indicators of the actual likelihood of a correct prediction than DNNs trained without mixup (see Figure 6.1 bottom row for an example). We also observe that merely mixing features does not result in the same calibration benefit and that the label smoothing in mixup training plays a significant role in improving calibration. Further, we also observe that mixup-trained DNNs are less prone to over-confident predictions on out-of-distribution and random-noise data.

The rest of this chapter is organized as follows: Section 6.2 provides a brief overview of the mixup training process; Section 6.3 discusses calibration metrics, experimental setup and mixup's calibration benefits for image data with additional results on natural language data described in Section 6.3.5; in Section 6.4, we explore in more detail the effect of mixup-based label smoothing on calibration, and further discuss the effect of training time on calibration in Section 6.3.7; in Section 6.5 we show additional evidence for the benefit of mixup training on predictive uncertainty when dealing with out-of-distribution data. Further discussions and conclusions are in Section 6.6.

6.2 An Overview of Mixup Training

Mixup training (Zhang et al., 2017) is based on the principle of Vicinal Risk Minimization (Chapelle et al., 2001)(VRM): the classifier is trained not only on the training data, but

also in the *vicinity* of each training sample. The vicinal points are generated according to the following simple rule introduced in (Zhang et al., 2017):

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j$$

where x_i and x_j are two randomly sampled input points, and y_i and y_j are their associated one-hot encoded labels. This has the effect of the empirical Dirac delta distribution

$$P_\delta(x, y) = \frac{1}{n} \sum_i^n \delta(x = x_i, y = y_i)$$

centered at (x_i, y_i) being replaced with the *empirical vicinal distribution*

$$P_\nu(\tilde{x}, \tilde{y}) = \frac{1}{n} \sum_i^n \nu(\tilde{x}, \tilde{y} | x_i, y_i) \delta(x = x_i, y = y_i)$$

where ν is a *vicinity distribution* that gives the probability of finding the virtual feature-target pair $(\tilde{x}, \tilde{y})$ in the vicinity of the original pair (x_i, y_i). The vicinal samples $(\tilde{x}, \tilde{y})$ are generated as above, and during training minimization is performed on the *empirical vicinal risk* using the vicinal dataset $\mathcal{D}_\nu := \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^m$:

$$R_\nu(f) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(f(\tilde{x}_i), \tilde{y}_i)$$

where L is the standard cross-entropy loss, but calculated on the soft-labels $\tilde{y}_i$ instead of hard labels. Training this way not only augments the feature set $\tilde{X}$, but the induced set of soft-labels also encourages the strength of the classification regions to vary linearly betweens samples. The experiments in (Zhang et al., 2017) and related work in (Inoue, 2018; Verma et al., 2018; Guo et al., 2018) show noticeable performance gains in various image classification tasks. The linear interpolator $\lambda \in [0, 1]$ that determines the mixing ratio is drawn from a

symmetric Beta distribution, $Beta(\alpha, \alpha)$ at each training iteration, where α is the hyper-parameter that controls the strength of the interpolation between pairs of images and the associated smoothing of the training labels. $\alpha = 0$ recovers the base case corresponding to zero-entropy training labels (in which case the resulting image is either just x_i or x_j), while a high value of α ends up in always averaging the inputs and labels. The authors in (Zhang et al., 2017) remark that relatively smaller values of $\alpha \in [0.1, 0.4]$ gave the best performing results for classification, while high values of α resulted in significant under-fitting. In this work, we also look at the effect of α on the calibration of the trained models.

6.3 Experiments

We perform numerous experiments to analyze the effect of mixup training on the calibration of the resulting trained classifiers on both image and natural language data. We experiment with various deep architectures and standard datasets, including large-scale training with ImageNet. In all the experiments in this paper, we only apply mixup to *pairs* of images as done in (Zhang et al., 2017). The mixup functionality was implemented using the mixup authors' code available at (Zhang,).

6.3.1 Setup

For the small-scale image experiments, we use the following datasets in our experiments: STL-10 (Coates et al., 2011), CIFAR-10 and CIFAR-100 (Krizhevsky and Hinton, 2009) and Fashion-MNIST (Xiao et al., 2017). For STL-10, we use the VGG-16 (Simonyan and Zisserman, 2014) network. CIFAR-10 and CIFAR-100 experiments were carried out on VGG-16 as well as ResNet-34 (He et al., 2016) models. For Fashion-MNIST, we used a ResNet-18 (He et al., 2016) model. For all experiments, we use batch normalization, weight decay of 5×10^{-4} and trained the network using SGD with Nesterov momentum, training for 200 epochs with an initial learning rate of 0.1 halved at 2 at 60,120 and 160 epochs. Unless otherwise noted, calibration results are reported for the best performing epoch on the validation set.

6.3.2 Calibration Metrics

We measure the calibration of the network as follows (and as described in (Guo et al., 2017)): predictions are grouped into M interval bins of equal size. Let B_m be the set of samples whose prediction scores (the winning softmax score) fall into bin m. The accuracy and confidence of B_m are defined as

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i)$$
$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i$$

where $\hat{p}_i$ is the confidence (winning score) of sample i. The **Expected Calibration Error** (ECE) is then defined as:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \Big| \text{acc}(B_m) - \text{conf}(B_m) \Big|$$

In high-risk applications, confident but wrong predictions can be especially harmful; thus we also define an additional calibration metric – the **Overconfidence Error** (OE) – as follows

$$\text{OE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \Big[\text{conf}(B_m) \times \max \Big(\text{conf}(B_m) - \text{acc}(B_m), 0 \Big) \Big]$$

This penalizes predictions by the weight of the confidence but only when confidence exceeds accuracy; thus overconfident bins incur a high penalty[1].

6.3.3 Comparison Methods

Since mixup produces smoothed labels over mixtures of inputs, we compare the calibration performance of mixup to two other label smoothing techniques:

---

[1]One could also similarly introduce an *under-confidence penalty*, but DNNs do not seem to be prone to such a phenomenon, so we do not explicitly report under-confidence in this work.

- $\epsilon-$*label smoothing* described in (Szegedy et al., 2016), where the one-hot encoded training signal is smoothed by distributing an ϵ mass over the other (i.e., non ground-truth) classes, and

- *entropy-regularized loss (ERL)* described in (Pereyra et al., 2017) that discourages the neural network from being over-confident by penalizing low-entropy distributions.

Our baseline comparison (no mixup) is regular training where no label smoothing or mixing of features is applied. We also note that in this section we do not compare against the temperature scaling method described in (Guo et al., 2017), which is a post-training calibration method and will generally produce well-calibrated scores. Here we would like to see the effect of label smoothing while training; experiments with temperature scaling are reported in Section 6.5.

6.3.4 Results

Results on the various datasets and architectures are shown in Figure 6.2. While the performance gains in validation accuracy are generally consistent with the results reported in (Zhang et al., 2017), here we focus on the effect of mixup on network calibration. The top row shows a calibration scatter plot for STL-10 and CIFAR-100, highlighting the effect of mixup training. In a well calibrated model, where the confidence matches the accuracy most of the points will be on $x = y$ line. We see that in the base case, both for STL-10 and CIFAR-100, most of the points tend to lie in the overconfident region. The mixup case is much better calibrated, noticeably in the high-confidence regions. The bar plots in the middle row provide results for accuracy and calibration for various combinations of datasets and architectures against comparison methods. We report the calibration error for the best performing model (in terms of validation accuracy). For label smoothing, an $\epsilon \in [0.05, 0.1]$ performed best while for ERL, the best-performing confidence penalty hyper-parameter was 0.1. The trends in the comparison are clear: label smoothing either via ϵ-smoothing, ERL or

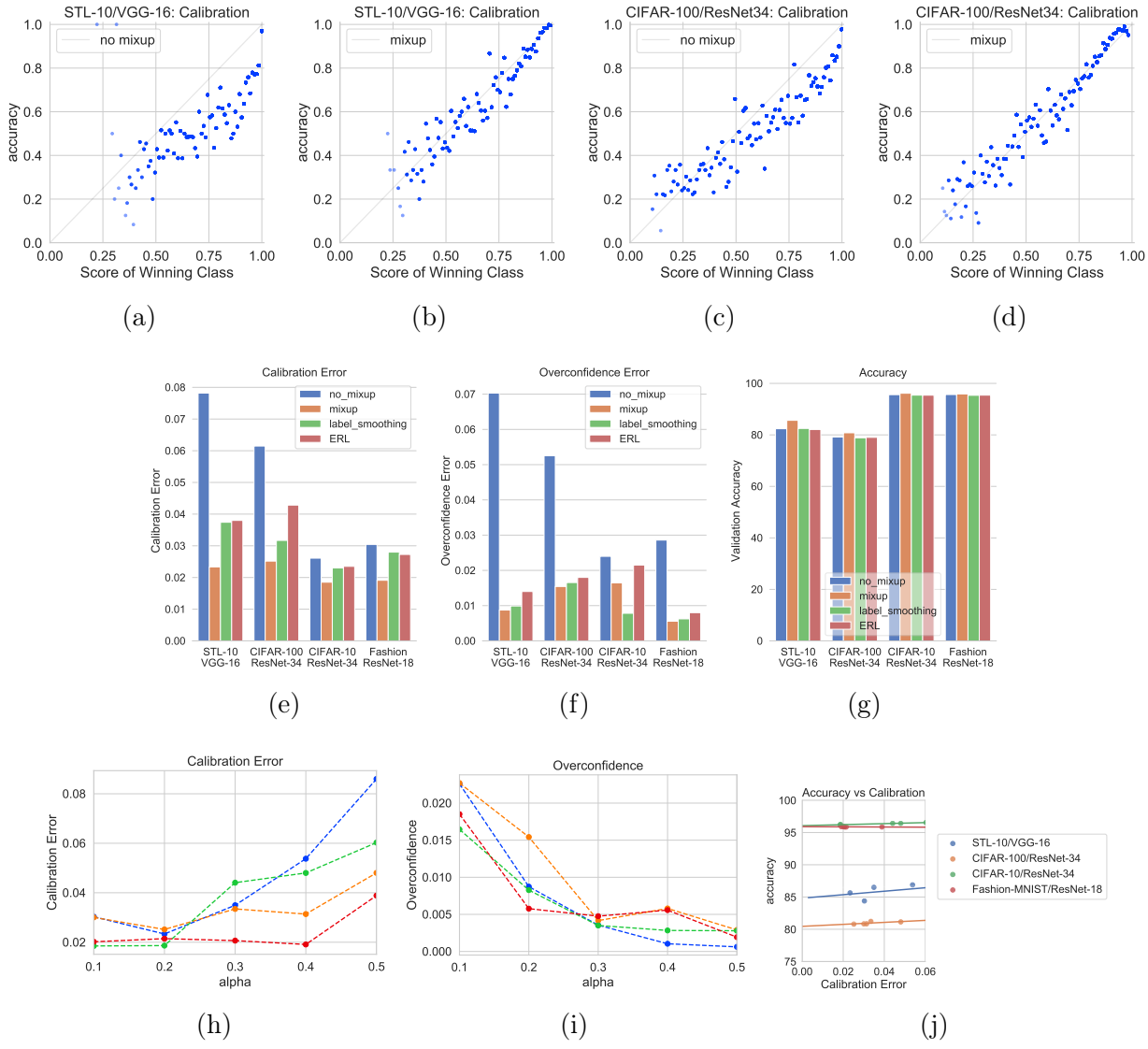
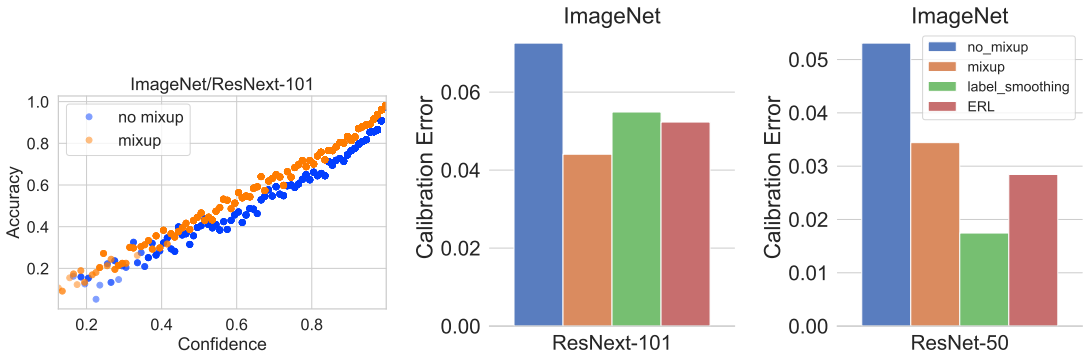


Figure 6.2: Calibration results for mixup and baseline (no mixup) on various image datasets and architectures. **Top Row**: Scatterplots for accuracy and confidence for STL-10(a,b) and CIFAR-100(c,d). The mixup case is much better calibrated with the points lying closer to the $x = y$ line, while in the baseline, points tend to lie in the overconfident region. **Middle Row:** Mixup versus comparison methods where label_smoothing is the ϵ-label smoothing method and ERL is the entropy regularized loss. **Bottom Row**: Expected calibration error (e) and overconfidence error (f) on various architectures. Experiments suggest best ECE is achieved for α in the [0.2,0.4] (h), while overconfidence error decreases monotonically with α due to under-fitting (i). Accuracy behavior for differently calibrated models is shown in (j).

mixup generally provides a calibration advantage and tempers overconfidence, with the latter generally performing the best in comparison to other methods.

We also show the effect on ECE as we vary the hyperparameter α of the mixing parameter distribution. For very low values of α, the behavior is similar to the base case (as expected), but ECE noticeably worsens for higher values of α due to the model being *under-confident*. For large α, the mixing parameter approaches 0.5, and thus the mixed-up image is an average of the input pair, and significantly different from either source. Since there are potentially $\mathcal{O}(n^2)$ such pairs, this results in under-fitting as the model is constantly seeing a steady stream of new images. Overconfidence alone decreases monotonically as we increase α as shown in Figure 6.2i. We also show the accuracy of mixup models at various levels of calibration determined by α. As can be seen, a well-tuned α can result in a better-calibrated model with very little loss in performance. Our classification results here are consistent with those reported in (Zhang et al., 2017) where the best performing α was in the $[0.1, .0.4]$ range.

Large-scale Experiments on ImageNet



(a) (b) (c)

Figure 6.3: Calibration on ImageNet for ResNet architectures

Here we report the results of calibration metrics resulting from mixup training on the 1000-class version of the ImageNet (Deng et al., 2009) data comprising of over 1.2 million images.

One of the advantages of mixup and its implementation is that it adds relatively little overhead to the training time (which is dominated by the computations for back-propagation), and thus can be easily applied to large scale datasets like ImageNet. We perform distributed parallel training using the synchronous version of stochastic gradient descent. We use the learning-rate schedule described in (Goyal et al., 2017) on a 32-GPU cluster and train till 93% accuracy is reached over the top-5 predictions. We test on two modern state-of-the-art archictures: ResNet-50 (He et al., 2016) and ResNext-101 (32x4d) (Xie et al., 2017). The results are shown in Figure 6.3. The scatter-plot showing calibration for ResNext-101 architecture suggests that mixup training provides noticeable benefits even in the large-data scenario, where the models should be less prone to over-fitting the one-hot labels. On the deeper ResNext-101, mixup provides better calibration than the label smoothing models, though this same effect was not visible for the ResNet-50 model. However, both calibration error and overconfidence show noticeable improvements using label smoothing over the baseline. The mixup model did however achieve a consistently higher classification performance of ≈ 0.4 percent over the other methods.

6.3.5 Experiments on Natural Language Data

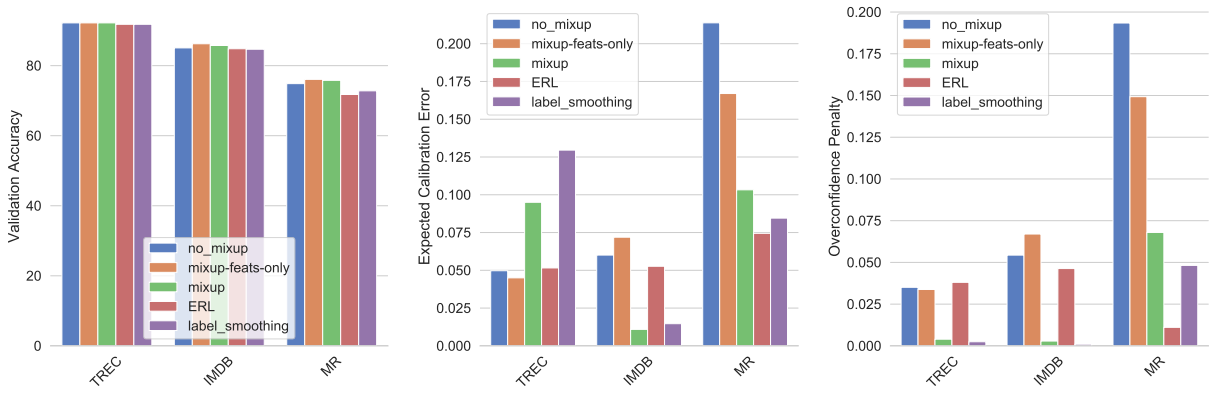


Figure 6.4: Accuracy, calibration and overconfidence on various NLP datasets

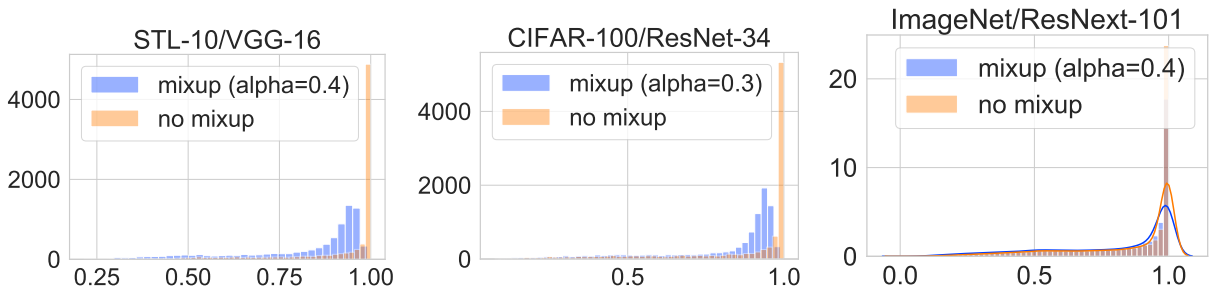
While mixup was originally suggested as a method to improve performance on image classification tasks, here we explore the effect of mixup training in the natural language processing (NLP) domain. A straight-forward mixing of inputs (as in pixel-mixing in images) will generally produce nonsense input since the semantics are unclear. To avoid this, we modify the mixup strategy to perform mixup on the embeddings layer rather than directly on the input documents. We note that this approach is similar to the recent work described in (Guo et al., 2019) that utilizes mixup for improving sentence classification which is among the few works, besides ours, studying the effects of mixup in the NLP domain. For our experiments, we employ mixup on NLP data for text classification using the MR (Pang and Lee, 2005), TREC (Li and Roth, 2002) and IMDB (Maas et al., 2011) datasets.We train a CNN for sentence classification (Sentence-level CNN) (Kim, 2014), where we initialize all the words with pre-trained GloVe (Pennington et al., 2014) embeddings, which are modified while training on each dataset. For the remaining parameters, we use the values suggested in (Kim, 2014). We refrain from training the most recent NLP models (Howard and Ruder, 2018; Cer et al., 2018; Zhou et al., 2016) since our aim here is not to show state-of-art classification performance on these datasets, but to study the effect on calibration. We show these results in Figure 6.4 where it is evident that mixup provides noticeable gains for all datasets, both in terms of calibration and overconfidence. We leave further exploration of principled strategies for mixup for NLP as future work.

6.3.6 Additional Experiments on Mixup Calibration

We also show the distribution of the winning scores for various image datasets, in Figure 6.5, where we see that mixup models are less peaked in the very-high confidence region.

For completeness, we also provide results using CIFAR-10 and CIFAR-100 on the ResNet-18 architecture as used in existing literature on mixup (Zhang et al., 2017; Verma et al., 2018; Guo et al., 2018), although these works did not consider the calibration aspect.

Results are shown in Table 6.1. The baseline performance (no mixup) matches the baseline accuracies reported in the previous literature. We also provide the expected calibration error



(a) (b) (c)

Figure 6.5: Distribution of winning scores on various image datasets

Method	**Test Accuracy**	**ECE**
No Mixup (Baseline)	95.12	0.023
Mixup (α=0.4)	96.16	0.019
Mixup (α=1.0)	96.04	0.1
Label Smoothing (ϵ=0.1)	95.51	0.089
ERL (κ=0.1)	95.55	0.046

(a) CIFAR-10/ResNet-18

Method	**Test Accuracy**	**ECE**
Baseline	78.28	0.049
Mixup (alpha=0.5)	79.57	0.035
Mixup (alpha=1.0)	79.54	0.091
Label Smoothing (eps=0.1)	79.08	0.066
ERL (kappa=1.0)	78.47	0.6

(b) CIFAR-100/ResNet-18

Table 6.1: Mixup results on CIFAR datasets with the ResNet 18 architecture

(ECE) for the best performing model as well as the mixup model that used $\alpha = 1.0$ that was used in the previous literature. We find that lower α gives slightly better classification and signficantly better ECE. Note that ECE can be high due to both the model being overconfident as well as under-confident, the latter being the case for $\alpha = 1.0$ since this causes the resulting training signal to have higher entropies than with smaller α's.

6.3.7 Effect of Extended Training on Mixup Calibration

As remarked in the previous section, one of the contributing factors to improved calibration in mixup is the significant data augmentation aspect of mixup training, where the model is unlikely to see the same mixed-up sample more than once. The natural question here is whether these models will eventually become overconfident if trained for much longer periods. In Figure 6.6, we show the training curves for a few extended training experiments where the models were trained for 1000 epochs: for the baseline (i.e when $\alpha = 0.$), the train loss and accuracy approach 0 and 100% respectively (i.e., over-fitting), while in the mixup case (non-zero α's), the strong data augmentation prevents over-fitting. This behavior is sustained over the entire duration of the training as can be seen in the corresponding values of ECE. Mixup models, even when trained for much longer, continue to have a low calibration error, suggesting that the mixing of data has a sustained inhibitive effect on over-fitting the training data (the training loss for mixup continues to be significantly higher than baseline even after extended training) and preventing the model from becoming overconfident.

6.3.8 Mixing in the Hidden Layers: Manifold Mixup and Effects on Calibration

The empirical results in this paper show that convexly combining features and labels significantly improves model calibration and predictive uncertainty of deep neural networks since the higher entropy training signal makes the model less confident in the regions of interpolated data. One natural extension to the basic mixup idea is *manifold mixup* proposed in (Verma et al., 2018), where the representations in the *hidden layers* are also combined linearly. The authors demonstrate that interpolation in hidden layers smooths the decision boundaries and encourages the model to learn class representations with fewer directions of variance. Here we emprically investigate the effect of this additional training signal from the hidden layers on model calibration. We use the PreActResNet architectures with the CIFAR-10 and CIFAR-100 datasets identical to those used in (Verma et al., 2018). We train the models for both 200 epochs using the same learning rate as we used in our earlier experiments, as

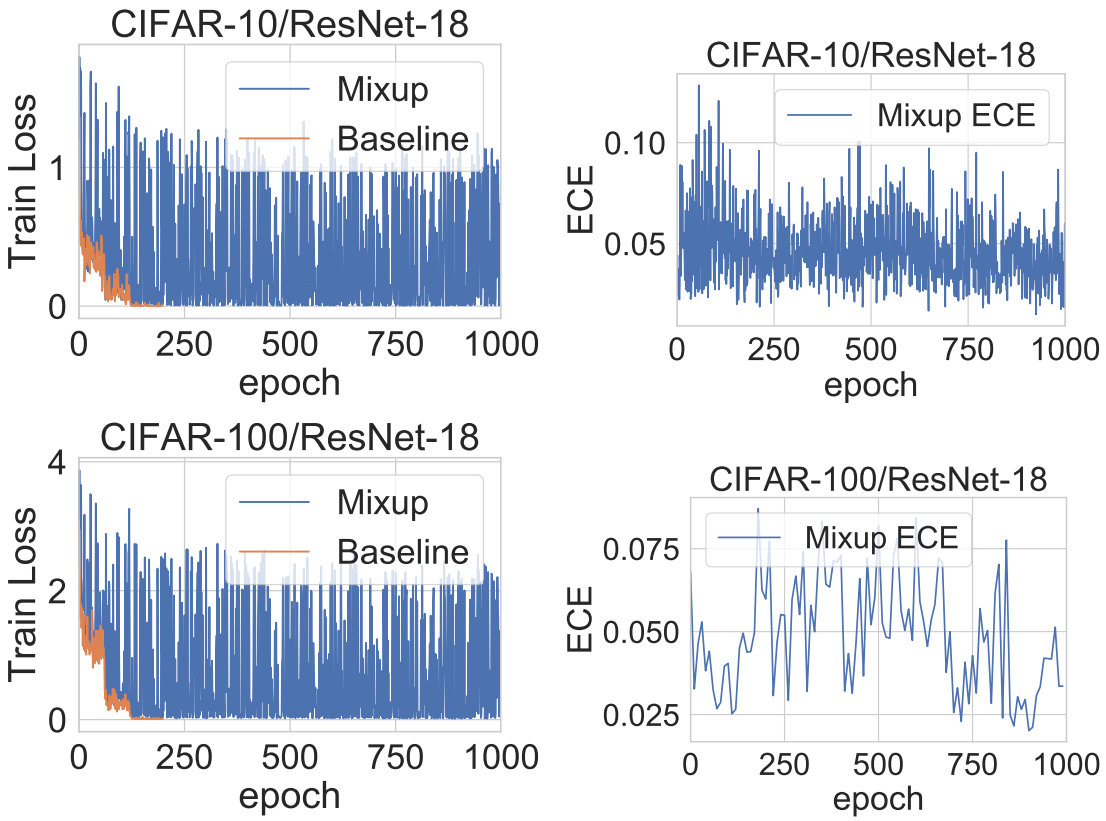


Figure 6.6: Training loss and calibration error under extended training for CIFAR-10 and CIFAR-100 with mixup. Baseline (no mixup) training loss (orange) goes to zero early on while mixup continues to have non-zero training loss even after 1000 epochs. Meanwhile, calibration error for mixup does not exhibit an upward trend even after extended training.

well as for 2000 epochs using the learning rate schedule in (Verma et al., 2018) and report on calibration and accuracy in both cases.

Results are in Table 6.2. When trained for the same number of epochs as the regular mixup experiments reported in this paper, manifold mixup generally has lower accuracy and worse calibration errors. However, accuracy is significantly better after training for 2000 epochs (as done in (Verma et al., 2018)), with ECE improving in a few cases. However, this is not a consistent trend, and further, since the manifold mixup algorithm is more complicated, involves more hyperparameters and takes longer to train than regular mixup, in practice,

Dataset	Method	Accuracy	ECE
CIFAR-10 (PreActResNet-18)	Mixup (200 epochs)	96.16	0.02
	Manifold Mixup (200 epochs)	95.96	0.047
	Manifold Mixup (2000 epochs)	97.07	0.077
CIFAR-10 (PreActResNet-34)	Mixup (200 epochs)	96.42	0.04
	Manifold Mixup (200 epochs)	96.2	0.04
	Manifold Mixup (2000 epochs)	97.5	0.007
CIFAR-100 (PreActResNet-18)	Mixup (200 epochs)	79.57	0.047
	Manifold Mixup (200 epochs)	74.79	0.22
	Manifold Mixup (2000 epochs)	79.73	0.06
CIFAR-100 (PreActResNet-34)	Mixup (200 epochs)	81.22	0.03
	Manifold Mixup (200 epochs)	77.36	0.187
	Manifold Mixup (2000 epochs)	82.32	0.03

Table 6.2: Manifold mixup experiments. For the extended training experiments, we use the same setup as (Verma et al., 2018) while for the experiments that trained for 200 epochs, we anneal the learning rate at epoch 60, 120 and 160 while keeping all other hyperparameters fixed to match the regular mixup experiments in Section 6.3

regular mixup might provide a more practical approach for improved calibration.

6.4 Effect of Soft Labels on Calibration

So far we have seen that mixup consistently leads to better calibrated networks compared to the base case, in addition to improving classification performance as has been observed in a number of works (Verma et al., 2018; Guo et al., 2018; Liang et al., 2018a). This behavior is not surprising given that mixup is a form of data augmentation: in mixup training, due to random sampling of both images as well as the mixing parameter λ, the probability that the learner sees the same image twice is small. This has a strong regularizing effect in terms of preventing memorization and over-fitting, even for high-capacity neural networks. Indeed, unlike regular training, the training loss in the mixup case is always significantly higher than the base case as observed by the mixup authors (Zhang et al., 2017). Because of the significant amount of data augmentation resulting from the random combination in mixup, from the perspective of statistical learning theory, the improved calibration of a mixup classifier can be

viewed as the classifier learning the true posteriors $P(Y|X)$ in the infinite data limit (Vapnik and Chervonenkis, 2015). However this leads to the following question: if the improved calibration is essentially an effect of data augmentation, does simply combining the images without combining the labels provide the same calibration benefit?

We perform a series of experiments on various image datasets and architectures to explore this question. Results from the earlier sections show that existing label smoothing techniques that increase the entropy of the training signal do provide better calibration without exploiting any data augmentation effects and thus we expect to see this effect in the mixup case as well. In the latter case, the entropies of the training labels are determined by the α parameter of the $Beta(\alpha, \alpha)$ distribution from which the mixing parameter is sampled. The distribution of training entropies for a few cases of α are shown in Figure 6.7. The base-case is equivalent to $\alpha = 0$ (not shown) where the entropy distribution is a point-mass at 0.

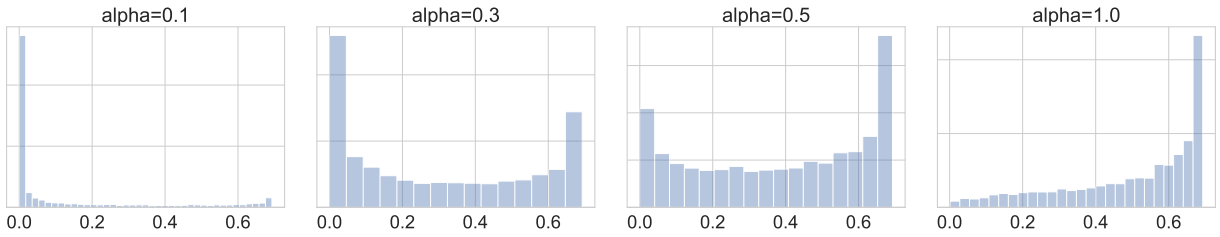


Figure 6.7: Entropy distribution of training labels as a function of the α parameter of the $Beta(\alpha, \alpha)$ distribution from which the mixing parameter is sampled.

To tease out the effect of full mixup versus only mixing features, we convexly combine images as before, but the resulting image assumes the hard label of the nearer class; this provides data augmentation without the label smoothing effect. Results on a number of benchmarks and architectures are shown in Figure 6.8. The results are clear: merely mixing features does not provide the calibration benefit seen in the full-mixup case suggesting that the point-mass distributions in hard-coded labels are contributing factors to overconfidence.

As in label smoothing and entropy regularization, having (or enforcing via a loss penalty) a non-zero mass in more than one class prevents the largest pre-softmax logit from becoming much larger than the others tempering overconfidence and leading to improved calibration.

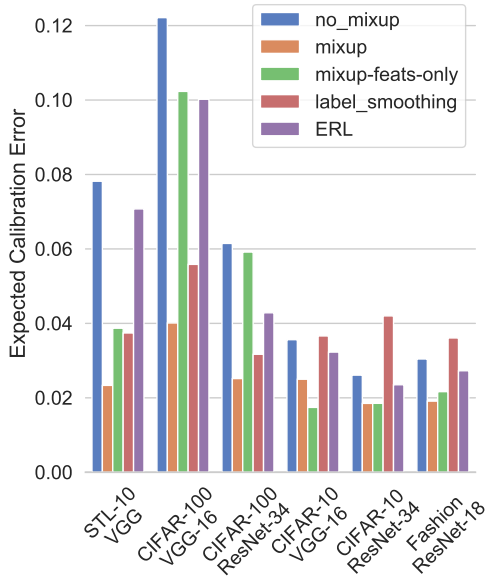


Figure 6.8: Calibration performance when only features are mixed vs. full mixup and other baselines on various datasets and architectures

6.5 Testing on Out-of-Distribution and Random Data

In this section, we explore the effect of mixup training when predicting on samples from unseen classes (out-of-distribution) and random noise images. Deep networks have been shown to produce pathologically overconfident predictions on random noise images (Hendrycks and Gimpel, 2016), and here we would like to explore the effect of mixup training on such behavior. We first train a VGG-16 network on in-distribution data (STL-10) and then predict on classes sampled from the ImageNet database that have not been encountered during training. For the random noise images, we test on gaussian random noise with the same mean and variance as the training set.

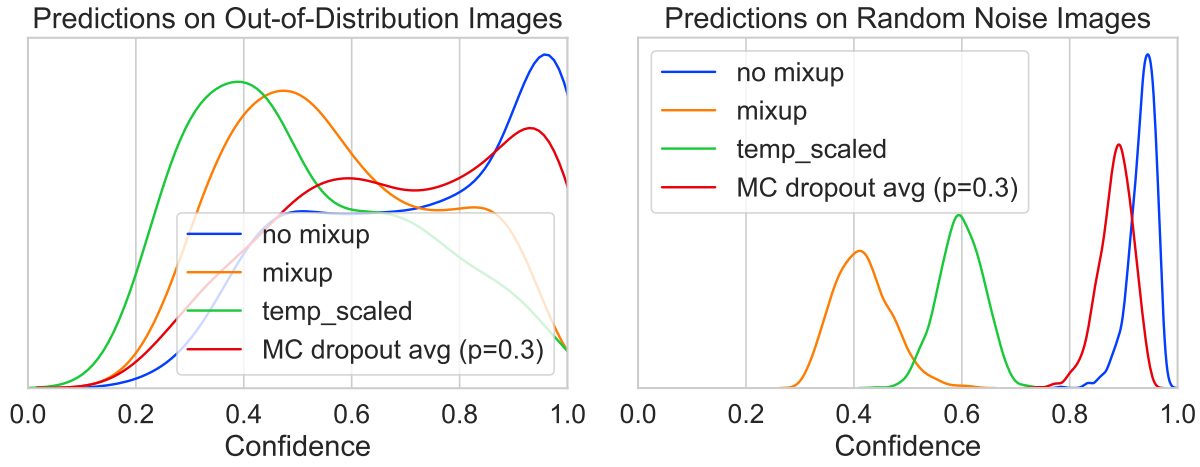


Figure 6.9: Distribution of winning scores from various models when tested on out-of-distribution and gaussian noise samples, after being trained on the STL-10 dataset.

We compare the performance of a mixup-trained model with that of the baseline, as well as a temperature calibrated pre-trained baseline as described in (Guo et al., 2017). Since the latter is a post-training calibration method, we expect it to be well-calibrated on in-distribution data. We also compare the prediction uncertainty using the Monte Carlo dropout method described in (Gal and Ghahramani, 2016) where multiple forward passes using dropout are made during test-time. We average predictions over 10 runs. The distribution over prediction scores for out-of-distribution and random data for mixup and comparison methods are shown in Figure 6.9. The differences versus the baseline are striking; in both cases, the mixup DNN is noticeably less confident than its non-mixup counterpart, with the score distribution being nearly perfectly separable in the random noise case. While temperature scaling is more conservative than mixup on real but out-of-sample data, it is noticeably more overconfident in the random-noise case. Further, mixup performs significantly better than MC-dropout in both cases.

In Table 6.3, we also show a comparison of the performance of the aforementioned models for reliably detecting out-of-distributon and random-noise data, using Area under the ROC (AUROC) curve as the metric. Mixup is the best performing model in both cases, significantly

Method	**AUROC(In/Out)**	
	STL-10/ ImageNet	STL-10/ Gaussian
Baseline	80.57	73.28
Mixup (α=0.4)	**83.28**	**95.93**
Temp. Scaling	56.2	54.2
Dropout(p=0.3)	78.93	70.57

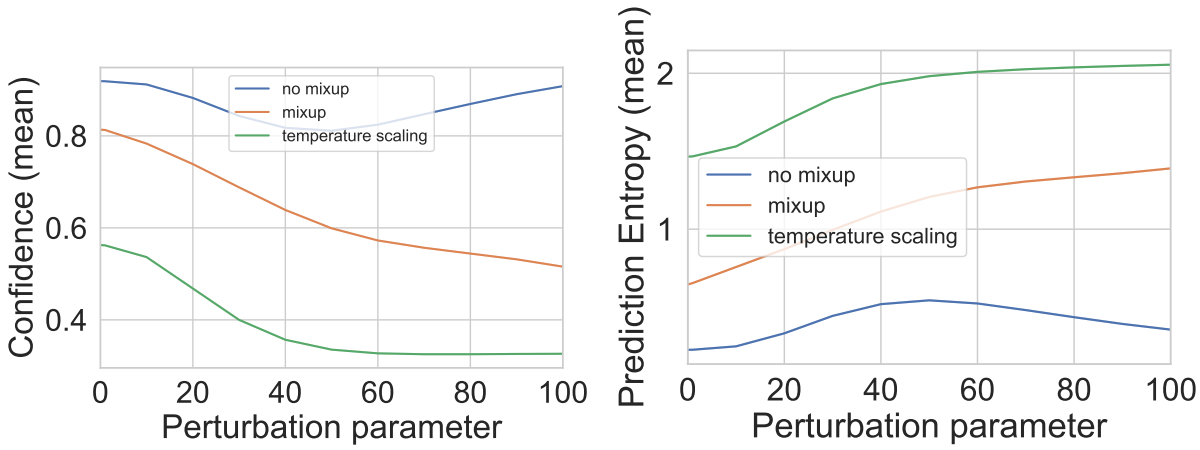
Table 6.3: Out-of-category detection results for the DAC on STL-10 and Tiny ImageNet.

outperforming the others as a random-noise detector. Temperature scaling, while producing well-calibrated models for in-distribution data is not a reliable detector. The scaling process reduces the confidence on both in and out-of-distribution data, significantly reducing the ability to discriminate between these two types of data. Mixup, on the other hand, does well in both cases. The results here suggest that the effect of training with interpolated samples and the resulting label smoothing tempers over-confidence in regions away from the training data. While these experiments were limited to two datasets and one architecture, the results indicate that training by minimizing vicinal risk can be an effective way to enhance reliability of predictions in DNNs. Note that since mixup trains the model by convexly combining pairs of images, the synthesized images all lie within the convex hull of the training data. In the next section, we provide results on the prediction confidence when images lie outside the convex hull of the training set.

6.5.1 Leaving the Convex Hull

Since mixup trains the model by convexly combining pairs of images, the synthesized images all lie within the convex hull of the training data. In this section, we explore the behavior of mixup as we gradually leave the convex hull in a random direction.

Specifically, given an input image $\mathbf{X} \in \mathbb{R}^m$, we choose a random vector $\mathbf{d} \in \mathbb{R}^m$ (where $d_i \sim U(-1, 1)$), and perturb $\mathbf{X}$ as follows: $\mathbf{X}' = \mathbf{X} + \mu\hat{\mathbf{d}}$. We try this for different $\mathbf{d}$'s and μ's and observe the predictions for a pre-trained mixup model and explore how the prediction



(a) (b)

Figure 6.10: Prediction behavior as one moves away from the training data

behavior changes.

We test three versions of a pre-trained VGG-16 model: mixup, baseline (no mixup) and a temperature-scaled version of the baseline, all trained on STL-10 data. We experiment over a wide range of the perturbation parameter μ. Figure 6.10 shows how the winning softmax scores (confidence) and the entropy of the prediction distributions change in all three cases.

When the images are perturbed and moved further away from the original manifold, temperature scaling quickly loses confidence as the perturbations get larger. Mixup confidence decays more gradually, while, remarkably, the base model loses confidence, and then quickly regains it as the images get further away from the training set – a pathological behavior of deep neural networks that has been widely observed in the literature. Threshold-based confidence models will obviously fail in such cases. Prediction entropy shows similar behavior to confidence. It is worthwhile to note that even a small perturbation of 0.01 (where the image structure is largely preserved) quickly degrades the confidence of temperature scaled models, indicating that they are less robust to additive noise; a threshold-based prediction mechanism will reject a significant number of samples in such cases.

6.6 Conclusion

We presented results on the effects of label smoothing – particularly the type of label soothing in mixup training – on calibration and predictive uncertainty. Existing empirical work has conclusively shown the benefits of mixup for boosting classification performance; in this work, we showed an additional important benefit: mixup trained networks, and also other label smoothing techniques, provide better calibrated and more reliable estimates both for in-sample and out-of-sample data.

There are possibly multiple reasons for this: the data augmentation provided by mixup is a form of regularization that prevents over-fitting and memorization, tempering overconfidence in the process. The label smoothing resulting from mixup might be viewed as a form of entropic regularization on the training signals, again preventing the DNN from driving the training error to zero. The results in this paper provide further evidence that training with hard labels is likely one of the contributing factors leading to overconfidence seen in modern neural networks. Recent work (Verma et al., 2018) has shown how the classification regions in mixup are smoother, without sudden jumps from one high confidence region to another suggesting that the lack of sharp transition boundaries in classification regions play an important role in producing well-calibrated classifiers.

Since mixup is implemented while training, it can also be employed with post-training calibration like temperature scaling, model perturbations like the dropout method or even the ensemble models described in (Lakshminarayanan et al., 2017). Further, mixup-based models can also be combined with rejection classifiers, both during training (as described in our work in Chapter 3) for dealing with label noise, as well as inference (Geifman and El-Yaniv, 2017) to improve the training and classification pipeline in modern deep learning. Indeed, the boost in classification performance coupled with the well-calibrated nature of mixup-trained DNNs as studied in this paper suggests that mixup based training, or even other label smoothing techniques such as $\epsilon-$smoothing, be employed in situations where predictive uncertainty is a significant concern.

Chapter 7

KNOWS WHEN IT DOESN'T KNOW: OPEN SET AND OUT-OF-DISTRIBUTION DETECTION USING ABSTENTION

Real knowledge is to know the extent of one's ignorance.

— *Confucius*

In this chapter, we attack the open set detection problem in image classification, where the DNN is presented with an object from an unknown category during test time, and has to be able to reliably recognize the sample as not being from any one of the training categories. Alternatively referred to in the literature as *out-of-distribution* or *novelty* detection, the ability to refrain from confidently predicting in such situations is an important trait for the safe deployment of deep learning systems. While simple to state, this has been challenging to solve reliably because, as discussed earlier, the peculiarities of DNNs make them prone to overconfident predictions. The work in this chapter demonstrates how the techniques we have developed so far in this thesis – abstention and improved calibration – can be used for reliable open set detection, improving upon other methods in the field.

7.1 Background

For most of the existence of machine learning, classification and recognition were treated as "closed set" scenarios: all classes that would be encountered at test time were assumed to be known at training time. However, the real-world is unpredictable and open-ended, and making such systems robust to the presence of unknown categories is essential for their safe deployment. A very active sub-field of machine learning, and now deep learning, known as *open set recognition*, or alternatively as *novelty* or *out-of-distribution* detection has emerged

in recent years that attempts to impart to deep neural networks the quality of "knowing when it doesn't know". While refraining from predicting when uncertain should be intuitively obvious to humans, the peculiarities of DNNs that we encountered in Chapter 5 makes this a challenging problem to solve in deep learning. In this chapter, we demonstrate the effectiveness of an abstention-based approach to attack this problem. We demonstrate how training with an extra class and augmenting the training set with samples from known outliers is a simple and highly effective technique for open set recognition and out-of-distribution detection that improves upon existing methods in the field; in the next section, we review some of these methods.

7.2 Related Work

Most of supervised machine learning has been developed with the "closed-set assumption"; in the event that unknown classes were encountered at test time, classifier scores were used to set rejection thresholds for determining the presence of unknown categories. Indeed, the ideas behind rejection classification that we discussed in Section 5.2 form the basis for most open set recognition (OSR) algorithms.

Because automatic speech recognition (ASR) systems were in deployment many years before computer vision systems became more commonplace, most of the early work in OSR came from the field of speech recognition. For example, the work in (Colton et al., 1995) describes a rejection-based approach to out-of-vocabulary words in an ASR system using neural networks, where the in-distribution samples were the utterances "male" and "female", and the out-of-distribution (OoD) samples were procured from utterances of names. In (Deng and Hu, 2003), an SVM is trained on the scores of a first-pass nearest neighbor classifier to determine known or unknown speakers. A similar application was considered in (Angkititrakul and Hansen, 2004) that first trains a Gaussian Mixture Model GMM, and then used a threshold-based rejection to determine if the speaker was from a known or unknown set.

With the advent of reliable computer vision systems, OSR was considered in the context

of face detection systems. A transductive approach is described in (Li and Wechsler, 2005) where confidence-thresholding based on distances computed from kNN classifiers was used to detect faces unknown to the system. The classification system for a real-time video-based face detection for entry access is described in (Stallkamp et al., 2007). Both kNN and GMM-based classifiers where used to output scores for nearest facial match; a threshold-based rejection model is then applied by training multiple 1-vs-rest classifiers. A recent open-set version of a kNN classifier was also described in (Júnior et al., 2017) that used the ratio of similarity scores to the two most similar classes; this was then combined with thresholding.

The problem of OSR was first mathematically formalized in (Scheirer et al., 2012) which introduced the concept of "Open Space Risk", defined formally as:

$$R_{\mathcal{O}}(f) = \frac{\int_{\mathcal{O}} f_y(x)dx}{\int_{S_o} f_y(x)dx} \tag{7.1}$$

where S_o is a ball in Euclidean space containing both the labeled open space $\mathcal{O}$ and all of the positive training examples, and f is the recognition function, where $f(x) = 1$ when a class is recognized and 0 otherwise. The authors proposed that the essential criterion of OSR was bounding the risk $R_{\mathcal{O}}(f)$, and noted that approaches like an SVM – that classify the data that are very far from any training sample – do not meet this criterion. However, the open space $\mathcal{O}$ was not formally defined. This was done in (Scheirer et al., 2014) that further extended the previous formalism and defined $\mathcal{O}$ as

$$\mathcal{O} = S_o - \bigcup_{i \in N} B_r\left(x_i\right) \tag{7.2}$$

where $B_r\left(x_i\right)$ is a closed ball of radius r centered around training sample x_i. The paper also introduced the "Compact Abating Probability" (CAP) model, which enforces the criterion that the recognition function decrease away from the training data; thresholding this then

bounds open space risk, such that for a given sample x and a known sample x_i,

$$\min_{x_i \in \mathcal{K}} \|x - x_i\| > \tau \Rightarrow M_{\mathrm{r}}(x) = 0 \tag{7.3}$$

where τ is the threshold and M_τ is the CAP model; an SVM calibrated with a Weibull distribution that satisfied the CAP model was then introduced for OSR.

The above ideas formed the basis of the work described in (Bendale and Boult, 2016) which considered OSR in the context of DNNs. Here a Weibull distribution is fit over the mean activation patterns of the known classes in the penultimate layer of the DNN, and an extra class is introduced that captures the mass of the unknown classes. Terming this approach "OpenMax" (as opposed to the standard softmax), the authors showed state-of-the-art performance in detecting OoD samples and nonsense images (Nguyen et al., 2015), and even for some types of adversarial perturbations. The OpenMax approach was also employed in (Shu et al., 2017) where multiple $1-$vs-rest sigmoidal classifiers were trained for OSR detection in text documents.

A simple baseline for detecting OoD samples using thresholded softmax scores was presented in (Hendrycks and Gimpel, 2016). While this approach is not conceptually new – thresholding on the winning class forms the basis of most rejection-based classifiers – the authors did provide empirical evidence that in many cases in deep learning, in-distribution predictions do tend to have higher winning scores than OoD samples, thus empirically justifying the use of softmax thresholding as a useful baseline. Another useful baseline is the temperature scaling method of (Guo et al., 2017) that was discussed in Section 5.3. While not explicitly defined for OSR, it does have the desirable property of producing calibrated scores by scaling down the softmax scores by a scalar T that is tuned on a validation set, namely:

$$\hat{q}_i = \max \sigma_{SM}\left(z_i / T\right) \tag{7.4}$$

which then justifies applying a threshold on the scores. While this produces calibrated scores

on in-distribution data, the ability to discriminate between in and out-of-distribution is hampered, as we saw in 6.5; nevertheless this approach can be useful when the OoD data is very unlike the in-distribution data, as shown in experiments from our work on *mixup* in Section 6.5. Indeed *mixup* itself, in addition to improving calibration, can also be used as a useful baseline for open set detection as shown in Figure 6.9.

Not surprisingly, methods developed for estimating uncertainty for DNN predictions have also been used for open set detection. For example the density estimation method described in (Subramanya et al., 2017), which estimates the density on pre-softmax activations and the "Deep Mahalanobis Detector" described in (Lee et al., 2018) – which fits a class conditional multi-variate Gaussian on the pre-softmax activations – can both be used for open set detection; like the earlier rejection-based methods, these involve density estimations using an already trained model. An ensemble-of-deep models approach, that is also augmented with adversarial examples during training, described in (Lakshminarayanan et al., 2017) was shown to improve predictive uncertainty and succesfully applied to OoD detection.

In the Bayesian realm, the works covered in Section 5.5 such as (Maddox et al., 2019) and (Osawa et al., 2019) have also been used for OoD detection, though at increased computational cost. However, it has been argued that for OSR, Bayesian priors on the data are not completely justified since one does not have access to the prior of the open-set (Boult et al., 2019). Nevertheless, simple approaches like dropout – which have been shown to be equivalent to deep gaussian processes (Gal and Ghahramani, 2016) – are also commonly used as baselines for OoD detection.

It has been observed that perturbing the input in the direction of increasing the confidence of the network's predictions on a given input has the effect of teasing apart in and out-of-distribution data. This was first exploited in the ODIN method described in (Liang et al., 2018b), which uses temperature scaling and a perturbation inspired by adversarial training. Concretely, given a trained neural network f, the temperature scaled softmax score is first

computed as:

$$S_i(\boldsymbol{x}; T) = \frac{\exp\left(f_i(\boldsymbol{x})/T\right)}{\sum_{j=1}^{N} \exp\left(f_j(\boldsymbol{x})/T\right)} \tag{7.5}$$

where T is the temperature scaling parameter. The input is then pre-processed to *increase* the softmax score as follows:

$$\overline{\boldsymbol{x}} = \boldsymbol{x} - \varepsilon \operatorname{sign}\left(-\nabla_x \log S_{\hat{\theta}}(\boldsymbol{x}; T)\right) \tag{7.6}$$

where ε is the perturbation magnitude. The overall approach is based on the observation that the magnitude of gradients on in-distribution data tend to be larger than for OoD data, and the perturbation thus enables separation between the two populations; state-of-the-art results were reported on many benchmarks. A drawback of this method, however, is that the hyperparameters T and ε need to be tuned on the OoD dataset, which is infeasible in many real-world scenarios as one does not often know in advance the properties of unknown classes. The method in (Lee et al., 2018) also involves perturbing the input in the direction of increased confidence score; confidence in this case, however, was measured by the Mahalanobis distance score using the computed mean and covariance of the pre-softmax scores.

The above methods all work with pre-trained models. In contrast, (DeVries and Taylor, 2018), describes a method inspired by ODIN and softmax thresholding, that used a separate confidence branch to train the DNN to additionally output a confidence score in its prediction. One of the contributions of this work was the observation that misclassified training data samples could serve as a useful proxy for OoD samples, and thus the method did not need additional data for negative samples. While this avoids an additional data gathering step, training the model to recognize unknown classes by using data from categories that do not overlap with classes of interest *has* been shown to be quite effective for out-of-distribution detection (Hendrycks et al., 2018a); here the predictions on the outliers used in training are regularized against the uniform distribution to encourage high-entropy posteriors on outlier samples.

Another method that incorporates OoD detection during training, and inspired by OpenMax, is described in (Yoshihashi et al., 2019) where, arguing that the mean activation vector approach in (Bendale and Boult, 2016) is insufficient for open-set loss, a method to jointly learn features for classification as well as reconstruction is described. The idea here is that training on the classification loss preserves in-distribution classification accuracy, while the reconstruction loss – which is expected to be high for unknown classes – enables the network to learn representations for discriminating between known and unknown classes.

Somewhat similar to the method we propose, an approach that uses an extra-class for outlier samples is described in (Neal et al., 2018), where instead of natural outliers, *counterfactual* images that lie just outside the class boundaries of known classes are generated using a GAN and assigned the extra class label. A similar approach using generative samples for the extra class, but using a conditional Variational Auto-Encoders (Kingma and Welling, 2013) for generation, is described in (Vernekar et al., 2019).

A method to force a DNN to produce high-entropy (i.e., low confidence) predictions and suppress the magnitude of feature activations for OoD samples was discussed in (Dhamija et al., 2018). Formally, the "Entropic Open Set Loss", defined as,

$$J_E(x) = \begin{cases} -\log S_c(x) & \text{if } x \in \mathcal{D}_{in} \text{ is from class } c \\ -\frac{1}{C}\sum_{c=1}^{C} \log S_c(x) & \text{if } x \in \mathcal{D}_{out} \end{cases} \tag{7.7}$$

where $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$ represent set of samples from known and unknown classes, and $S_c(x)$ is the softmax output for sample x. The above formulation encourages a uniform output distribution over all known classes when $x \in \mathcal{D}_{out}$. Arguing that methods that use an extra background class for OoD samples force all such samples to lie in one region of the feature space, the work also forces separation by suppressing the activation magnitudes of samples from unknown classes by adding an "Objectosphere" loss term as follows:

$$J_R = J_E + \lambda \begin{cases} \max(\xi - \|F(x)\|, 0)^2 & \text{if } x \in \mathcal{D}_{in} \\ \|F(x)\|^2 & \text{if } x \in \mathcal{D}_{out} \end{cases} \tag{7.8}$$

where $F(x)$ is the feature vector that feeds into the pre-softmax (logit) layer of the DNN. This forces the activation vectors from known-class samples to lie outside a $\xi-$ball around the origin, while encouraging feature representations of unknown class samples to lie closer to the origin. Scores are then thresholded using a confidence measure that is defined as $c = S_c(x)||F(x)||$. In the experiments, OoD data were sampled from real images of non-overlapping classes; results were shown on various standard image benchmark datasets and DNN architectures.

The abstention method that we describe in the next section borrows ideas from many of the above methods. It uses additional samples of *natural* images from non-overlapping categories to train the model to abstain, but using an extra abstention class. While some of the work described previously, such as (Dhamija et al., 2018) argue that using outliers and an additional class are not very effective, comprehensive experiments we conduct in this chapter demonstrate that this is not the case. We will see that the presence of an abstention class, and augmenting the training set with outliers can be a highly effective approach for OoD and open set detection; we describe the details in the next section.

7.3 Out-of-Distribution Detection with the DAC

Our approach uses the Deep Abstaining Classifier (DAC) for detecting out-of-distribution and novel samples, but unlike the DAC-based approach introduced in Chapter 3 for tackling label noise, we make two significant modifications: first, instead of the DAC loss in Equation 3.1, here we use regular cross entropy for training; and second, we augment our training set of in-distribution samples ($\mathcal{D}_{in}$) with an auxiliary dataset of *known* out-of-distribution samples ($\tilde{\mathcal{D}}_{out}$), that are known to be mostly disjoint from the main training set (we will use $\mathcal{D}_{out}$ to denote *unknown* out-of-distribution samples that we use for testing). We assign the training label of $K+1$ to all the outlier samples in $\tilde{\mathcal{D}}_{out}$ (where K is the number of known classes); the minimization problem then becomes:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{in}} \left[-\log P_\theta(y=\hat{y}|\mathbf{x})\right] + \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\tilde{\mathcal{D}}_{out}} \left[-\log P_\theta(y=K+1|\mathbf{x})\right] \tag{7.9}$$

where θ are the weights of the neural network. This is somewhat similar to the approaches described in (Hendrycks et al., 2018a) as well as in (Lee et al., 2017), with the main difference being that in those methods, an extra class is not used; instead predictions on outliers are regularized against the uniform distribution. That is, the minimization in their formulation is:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{in}} \left[-\log P_\theta(y=\hat{y}|\mathbf{x})\right] + \lambda\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\tilde{\mathcal{D}}_{out}} \left[KL\left(\mathcal{U}(y)\|P_\theta(y|\mathbf{x})\right)\right] \tag{7.10}$$

The approach in (Hendrycks et al., 2018a) uses an outlier set of natural images as $\tilde{\mathcal{D}}_{out}$, while (Lee et al., 2017) uses a generative adverarial network (GAN) (Goodfellow et al., 2014a) to generate samples in $\tilde{\mathcal{D}}_{out}$, making the latter method harder to train. Also, note that the loss on the outlier samples is weighted by a hyperparameter λ which has to be tuned; in contrast, our approach does not introduce any additional hyperparameters.

The use of known OoD samples (or outliers) has been shown to generalize to unknown samples, as demonstrated in (Hendrycks et al., 2018a), and also in other works we described in the earlier section, such as (Neal et al., 2018) and (Vernekar et al., 2019). Another example of such an approach is described in (Ge et al., 2017) which extends the idea of OpenMax from (Bendale and Boult, 2016), but unlike the latter, which works with a pre-trained model, their method uses a GAN to generate synthetic samples for the extra class by sampling from mixture distributions of known classes. Thus, even though the space of unknown classes is potentially infinite, and one can never know in advance the myriad of inputs that can occur during test time, empirically this approach has been shown to work and hence we adopt it in our work. We find that the presence of an abstention class that is used to capture the mass in $\tilde{\mathcal{D}}_{out}$ significantly increases the ability to discriminate between in and out-of-distribution

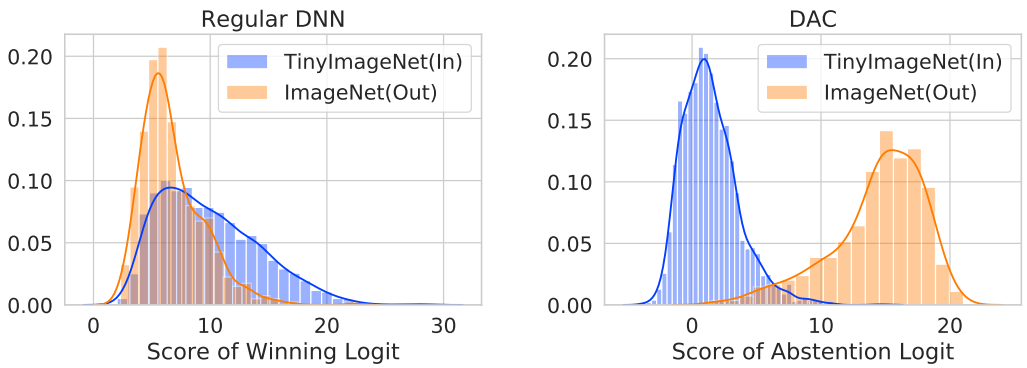


Figure 7.1: An illustration of the separability of scores on in and out-of-distribution data for a regular DNN (left) and the Deep Abstaining Classifier (DAC) (Right); the presence of the abstention class enables the DAC to produce higher separation and thus, better detection, on OoD samples.

data. For example, in Figure 7.1, we show the distribution of the winning logits (pre-softmax activations) in a regular DNN (left). For the same experimental setup, the *abstention logit* of the DAC produces near-perfect separation of the in and out-of-distribution logits indicating that using an abstention class for mapping outliers can be a very effective approach to OoD detection.

Once trained, we use a simple thresholding mechanism for detection. Concretely, the detector, $g(x) : X \rightarrow 0, 1$ assigns label 1 (OoD) if the softmax score of the abstention class, i.e., $p_{K+1}(x)$ is above some threshold δ, and label 0, otherwise:

$$g(\mathbf{x}) = \begin{cases} 1 & \text{if } p_{K+1}(x) \geq \delta \\ 0 & \text{otherwise} \end{cases} \tag{7.11}$$

Like in other methods, the threshold δ has to be determined based on acceptable risk that might be specific to the application. However, using performance metrics like area under the ROC curve (AUROC), we can determine threshold-independent performance of various methods, and we use this as one of our evaluation metrics in all our experiments.

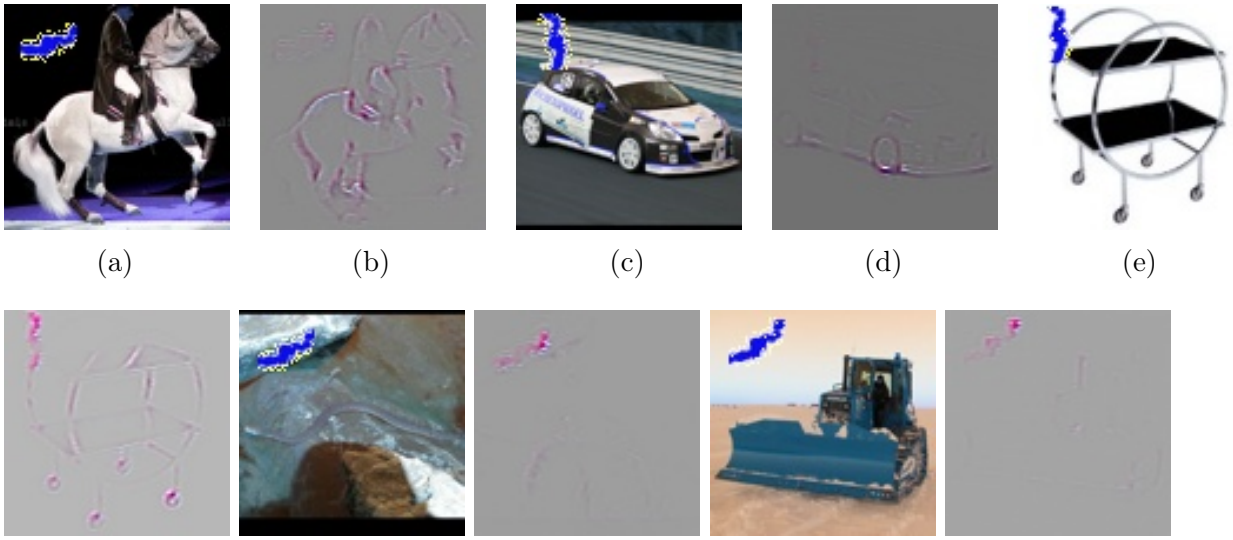


Figure 7.2: Images and filter visualizations illustrating the usage of DAC as an out-of-category detector. During inference, all incoming images are augmented with a fixed feature (a smudge for instance, shown as a blue streak here.) When presented with known classes, (horse (a) or car (c)), the activations for the class features are much more salient than the smudge (b,d). For objects from unknown classes (e,g,i), the absence of known features causes the smudge to be the most salient feature (f,h,j) resulting in abstention.

As an additional detection mechanism, we could also exploit a feature-based mechanism for OoD detection. The experimental results from Chapter 3 indicate that, depending on image content, there is context-dependent suppression of features based on what associations have been made during training. For example, from the filter visualizations in the smudging experiments (Figure 3.5), when the DAC is presented with the image of a cat without a smudge, the cat features are expressed clearly in the filters, while the same features get suppressed in the presence of the smudge since the smudge was strongly indicative of abstention.

In those experiments, the smudge was a feature encountered while training that became associated with the abstention class (due to label noise). By applying this feature to all the samples in $\tilde{\mathcal{D}}_{out}$, and *some fraction* q of samples in $\mathcal{D}_{in}$, the presence of a fixed known feature might be used to discriminate ID and OoD samples. Filter visualizations illustrating

the effect of training using such an approach are shown in Figure 7.2 (where we re-use the smudge here as our fixed feature). When presented with ID data that is also smudged (horse or car), the DAC activations for the class features are much more salient than the smudge. On the other hand, for OoD data, the absence of known features causes the smudge to be the most salient feature, driving the prediction to the abstention class.

The challenge here is that the fraction q of samples is a hyperparameter that has to be carefully determined on a dataset-specific basis to obtain the correct behavior. Further, a fixed feature (like a smudge) might be a legitimately occurring feature in a large dataset, and thus one would also have to determine an extraneous feature that does not interfere with the classification process. While this approach is promising, it introduces additional complexities in the training process. To keep things simple, the rest of this chapter will only describe results without using the feature augmentation approach.

7.4 Experiments

The experiments we describe here can be divided into two categories: in the first category, we compare against methods that are explicitly designed for OoD detection and open set recognition, while in the second category, we compare against methods that are known to improve predictive uncertainty in deep learning. In both cases, we report results over a variety of architectures and datasets to demonstrate the efficacy of our method.

7.4.1 Comparison against OoD and Open-Set Methods

In this section, we compare against a slew of recent state-of-the-art methods that are designed for OoD and novelty detection; we compare against the following:

- **Deep Anomaly Exposure**, as described in (Hendrycks et al., 2018a) and discussed in the previous section.

- **Ensemble of Leave-out Classifiers** (Vyas et al., 2018) where each classifier is trained

by leaving out a random subset of training data (which is treated as OoD data), and the rest is treated as ID data.

- **ODIN**, as described in (Liang et al., 2018b) and discussed in Section 7.2. ODIN uses input perturbation and temperature scaling to differentiate between ID and OoD samples, and is currently one of the state-of-the-art methods in OoD detection.

- **Deep Mahalanobis Detector**, proposed in (Lee et al., 2018) which estimates the class-conditional distribution over hidden layer features of a deep model using Gaussian discriminant analysis and a Mahalanobis distance based confidence-score for thresholding, and further, similar to ODIN, uses input perturbation while testing.

- **OpenMax**, as described in (Bendale and Boult, 2016) and discussed earlier in Section 7.2. This method uses mean activation vectors of ID classes observed during training followed by Weibull fitting to determine if a given sample is novel or out-of-distribution.

For the above methods, we use published results when available, keeping the architecture and datasets the same as in the experiments described in the respective papers. For OpenMax, we re-implement the authors' published algorithm using the PyTorch framework (Paszke et al., 2019).

Datasets

For the experiments in this section, we use CIFAR-10 and CIFAR-100 as the in-distribution datasets, in addition to augmenting our training set with 100K unlabeled samples from the Tiny Images dataset (Torralba et al., 2008); this was the same OoD dataset used in (Hendrycks et al., 2018a) and shown to give good results. For the out-of-distribution datasets, we test on the following:

- **SVHN** (Netzer et al., 2011), a large set of 32×32 color images of house numbers, comprising of ten classes of digits $0 - 9$. We use a subset of the 26K images in the test

set.

- **LSUN** (Yu et al., 2015), the Large-scale Scene Understanding dataset, comprising of 10 different types of scenes.

- **Places365** (Zhou et al., 2017), a large collection of pictures of scenes that fall into one of 365 classes.

- **Tiny ImageNet** (tin, 2017) (not to be confused with Tiny Images) which consists of images belonging to 200 categories that are a subset of ImageNet categories. The images are 64×64 color, which we scale down to 32×32 when testing.

- **Gaussian** A synthetically generated dataset consisting of 32×32 random Gaussian noise images, where each pixel is sampled from an i.i.d Gaussian distribution.

Metrics

Following established practices in the literature, we use the following metrics to measure detection performance of our method:

- **AUROC** or Area Under the Receiver Operating Characteristic curve depicts the relationship between the True Positive Rate (TPR) (also known as Recall)and the False Positive Rate (FPR) and can be interpreted as the probability that a positive example is assigned a higher detection score than a negative example (Fawcett, 2006). Each point on the ROC is generated by applying a threshold on the scores and computing TPR and FPR at that threshold. A perfect detector corresponds to an AUROC of 1, while a purely random classifier corresponds to an AUROC of 0.5. Unlike 0/1 accuracy, the AUROC has the desirable property that it is not affected by class imbalance[1].

[1]An alternate area-under-the-curve metric, known as Area under Precision Recall Curve, or AUPRC, is used when the size of the negative class is high compared to the positive class. We do not report AUPRC here, as we keep our in-distribution and out-of-distribution sets balanced in these experiments.

- **FPR at 95% TPR** which is the probability that a negative sample is misclassified as a positive sample when the TPR (or recall) on the positive samples is 95%.

In work that we compare against, the out-of-distribution samples are treated as the positive class, so we do the same here, and treat the in-distribution samples as the negative class.

Results

Detailed results against the various OoD methods are shown in Table 7.1, where we have a clear trend: in almost all cases, the DAC outperforms the other methods, often by significant margins especially when the in-distribution data is more complex, as is the case with CIFAR-100. While the Outlier Exposure method (Hendrycks et al., 2018a) (shown at the top in Table 7.1) is conceptually similar to ours, the presence of an extra abstention class in our model often bestows significant performance advantages. Further, we do not need to tune a separate hyperparameter which determines the weight of the outlier loss, as done in (Hendrycks et al., 2018a).

In fact, the simplicity of our method is one of its striking features: we do not introduce any additional hyperparameters in our approach, which makes it significantly easier to implement than methods such as ODIN and the Mahalanobis detector; these methods need to be tuned separately on each OoD dataset, which is usually not possible as one does not have access to the distribution of unseen classes in advance. Indeed, when performance of these methods is tested without tuning on the OoD test set, the DAC significantly outperforms methods such as the Mahalanobis detector (shown at the bottom of Table 7.1). We also show the performance against the OpenMax approach of (Bendale and Boult, 2016) in Table 7.2 and in every case, the DAC outperforms OpenMax by significant margins.

While the abstention approach uses an extra class and OoD samples while training, and thus does incur some training overhead, it is significantly less expensive during test time, as the forward pass is no different from that of a regular DNN. In contrast, methods like ODIN

vs. Outlier Exposure (OE) (Hendrycks et al., 2018a)
(Model: Wide ResNet 40x2)

	$\mathcal{D}_{in}$: CIFAR-10				$\mathcal{D}_{in}$: CIFAR-100			
	FPR95 ↓		AUROC ↑		FPR95 ↓		AUROC ↑	
$\mathcal{D}_{out}$	OE	Ours	OE	Ours	OE	Ours	OE	Ours
SVHN	4.8	$\mathbf{2.0}_{0.69}$	98.4	$\mathbf{99.46}_{0.16}$	**42.9**	$\mathbf{40.46}_{11.96}$	**86.9**	$\mathbf{85.44}_{6.25}$
LSUN	12.1	$\mathbf{0.1}_{0.06}$	97.6	$\mathbf{99.96}_{0.02}$	57.5	$\mathbf{9.27}_{4.62}$	83.4	$\mathbf{97.67}_{1.40}$
Places365	17.3	$\mathbf{0.22}_{0.12}$	96.2	$\mathbf{99.92}_{0.05}$	49.8	$\mathbf{23.37}_{5.30}$	86.5	$\mathbf{93.98}_{1.67}$
Gaussian	0.7	$\mathbf{0.13}_{0.14}$	99.6	$\mathbf{99.93}_{0.09}$	**12.1**	$\mathbf{13.26}_{14.74}$	**95.7**	$\mathbf{90.03}_{11.81}$

vs. Ensemble of Leave-out Classifiers (ELOC) (Vyas et al., 2018)
(Model: Wide ResNet 28x10)

	$\mathcal{D}_{in}$: CIFAR-10				$\mathcal{D}_{in}$: CIFAR-100			
	FPR95 ↓		AUROC ↑		FPR95 ↓		AUROC ↑	
$\mathcal{D}_{out}$	ELOC	Ours	ELOC	Ours	ELOC	Ours	ELOC	Ours
Tiny ImageNet	**2.94**	$\mathbf{1.91}_{2.24}$	**99.36**	$\mathbf{99.45}_{0.67}$	**24.53**	$\mathbf{18.68}_{6.31}$	**95.18**	$\mathbf{94.88}_{1.75}$
LSUN	**0.88**	$\mathbf{1.5}_{1.80}$	**99.7**	$\mathbf{99.61}_{0.47}$	16.53	$\mathbf{9.23}_{1.87}$	96.77	$\mathbf{97.89}_{0.48}$
Gaussian	**0.0**	$\mathbf{0.13}_{0.20}$	99.58	$\mathbf{99.95}_{0.08}$	98.26	$\mathbf{0.72}_{0.79}$	93.04	$\mathbf{99.65}_{0.39}$

vs. ODIN (Liang et al., 2018b)
(Model: Wide ResNet 28x10)

	$\mathcal{D}_{in}$: CIFAR-10				$\mathcal{D}_{in}$: CIFAR-100			
	FPR95 ↓		AUROC ↑		FPR95 ↓		AUROC ↑	
$\mathcal{D}_{out}$	ODIN	Ours	ODIN	Ours	ODIN	Ours	ODIN	Ours
Tiny ImageNet	25.5	$\mathbf{1.91}_{2.24}$	92.1	$\mathbf{99.45}_{0.67}$	55.9	$\mathbf{18.68}_{6.31}$	84.0	$\mathbf{94.88}_{1.75}$
LSUN	17.6	$\mathbf{1.5}_{1.80}$	95.4	$\mathbf{99.61}_{0.47}$	56.5	$\mathbf{9.23}_{1.87}$	86.0	$\mathbf{97.89}_{0.48}$
Gaussian	**0.0**	$\mathbf{0.13}_{0.20}$	**100.0**	$\mathbf{99.95}_{0.08}$	**1.0**	$\mathbf{0.72}_{0.79}$	98.5	$\mathbf{99.65}_{0.39}$

vs. Deep Mahalanobis Detector (MAH) (Lee et al., 2018)
(Model: ResNet 34)

	$\mathcal{D}_{in}$: CIFAR-10				$\mathcal{D}_{in}$: CIFAR-100			
	FPR95 ↓		AUROC ↑		FPR95 ↓		AUROC ↑	
$\mathcal{D}_{out}$	MAH	Ours	MAH	Ours	MAH	Ours	MAH	Ours
SVHN	24.2	$\mathbf{1.89}_{0.78}$	95.5	$\mathbf{99.49}_{0.17}$	58.1	$\mathbf{41.31}_{8.01}$	84.4	$\mathbf{86.85}_{2.38}$
Tiny ImageNet	4.5	$\mathbf{0.36}_{0.15}$	99.0	$\mathbf{99.88}_{0.04}$	29.7	$\mathbf{12.10}_{1.22}$	87.9	$\mathbf{97.14}_{0.29}$
LSUN	1.9	$\mathbf{0.30}_{0.12}$	99.5	$\mathbf{99.91}_{0.03}$	43.4	$\mathbf{7.14}_{0.66}$	82.3	$\mathbf{98.45}_{0.13}$

Table 7.1: Comparison of the extra class method (ours) with various other out-of-distribution detection methods when trained on CIFAR-10 and CIFAR-100 and tested on other datasets. All numbers from comparison methods are sourced from their respective original publications. For our method, we also report the standard deviation over five runs (indicated by the subscript), and treat the performance of other methods within one standard deviations as equivalent to ours. For fair comparison with the Mahalanobis detector (MAH) (Lee et al., 2018), we use results when their method was not tuned separately on each OoD test set (Table 6 in (Lee et al., 2018).

vs. OpenMax (Bendale and Boult, 2016)
(Model: ResNet 34)

	$\mathcal{D}_{in}$: CIFAR-10				$\mathcal{D}_{in}$: CIFAR-100			
	FPR95 ↓		AUROC ↑		FPR95 ↓		AUROC ↑	
$\mathcal{D}_{out}$	OpenMax	Ours	OpenMax	Ours	OpenMax	Ours	OpenMax	Ours
SVHN	$23.67_{2.06}$	$\mathbf{1.89}_{0.78}$	$90.72_{0.90}$	$\mathbf{99.49}_{0.17}$	$53.22_{7.52}$	$\mathbf{41.31}_{8.01}$	$80.88_{1.08}$	$\mathbf{86.85}_{2.38}$
Tiny ImageNet	$24.20_{9.11}$	$\mathbf{0.36}_{0.15}$	$93.39_{0.75}$	$\mathbf{99.88}_{0.04}$	$32.67_{5.21}$	$\mathbf{12.10}_{1.22}$	$81.22_{2.21}$	$\mathbf{97.14}_{0.29}$
LSUN	$18.68_{1.24}$	$\mathbf{0.30}_{0.12}$	$92.16_{1.82}$	$\mathbf{99.91}_{0.03}$	$30.21_{2.71}$	$\mathbf{7.14}_{0.66}$	$83.08_{2.16}$	$\mathbf{98.45}_{0.13}$
Places365	$27.27_{2.77}$	$\mathbf{0.84}_{0.28}$	$90.72_{0.85}$	$\mathbf{99.67}_{0.07}$	$50.71_{1.25}$	$\mathbf{30.54}_{2.26}$	$81.13_{0.30}$	$\mathbf{92.69}_{0.65}$
Gaussian	$40.58_{22.18}$	$\mathbf{0.04}_{0.02}$	$84.74_{10.19}$	$\mathbf{99.98}_{0.01}$	$21.50_{11.73}$	$\mathbf{1.66}_{1.76}$	$89.37_{5.46}$	$\mathbf{99.48}_{0.47}$

Table 7.2: DAC vs OpenMax. The OpenMax implementation was based on code available at `https://github.com/abhijitbendale/OSDN` and re-implemented by us in PyTorch (Paszke et al., 2019).

and the Mahalanobis detector require gradient calculation with respect to the input in order to apply input perturbation. In summary, the performance and simplicity of our method makes this a powerful and practical approach when compared to existing work.

7.4.2 Comparison against Uncertainty-based Methods

Next we perform experiments to compare the OoD detection performance of the DAC against various methods that have been proposed for improving predictive uncertainty in deep learning. In these cases, one expects that such methods will cause the DNN to predict with less confidence when presented with inputs from a different distribution or from novel categories; we compare against the following methods:

- **Softmax Thresholding** This is the simplest baseline, where OoD samples are detected by thresholding on the winning softmax score; scores falling *below* a threshold are rejected. While simple, it has shown to be an effective baseline for OoD detection (Hendrycks and Gimpel, 2016).

- **Entropy Thresholding** Another simple baseline, where OoD samples are rejected if the Shannon entropy calculated over the softmax posteriors is *above* a certain threshold.

- **MonteCarlo Dropout** A Bayesian inspired approach proposed in (Gal and Ghahramani, 2016) for improving the predictive uncertainty for deep learning, and discussed in Section 5.5. The idea here is that dropout – where one randomly disables DNN connections during the forward pass, and often used as a regularization method in training (Srivastava et al., 2014) – is also enabled during *test* time. Multiple forward passes are made for predicting on a sample, with a randomly chosen fraction p of the network connections being de-activated in each pass; the final prediction is then an average over the ensemble. We found $p = 0.5$ to perform well, and use 100 forward passes per sample during the prediction; while expensive, this has been to shown to improve predictive uncertainty in deep models.

- **Temperature Scaling**, which improves DNN calibration as described in (Guo et al., 2017), and discussed in Section 7.2 and Chapters 5 and 6. The scaling temperature T is tuned on a held-out subset of the validation set of the in-distribution data.

- **Mixup** As shown in Chapter 6, and particularly in Section 6.5, Mixup can be an effective OoD detector, so we also use this as one of our baselines.

- **Deep Ensembles** which was introduced in (Lakshminarayanan et al., 2017) for improving uncertainty estimates for both classification and regression. In this approach, multiple versions of the same model are trained using different random initializations, and while training, adversarial samples are generated to improve model robustness. While computationally expensive, the results in (Lakshminarayanan et al., 2017) show that this can be an effective approach for improving uncertainty in deep learning; we use an ensemble size of 5 as suggested in their paper.

- **SWAG**, as described in (Maddox et al., 2019), which is another Bayesian approach to deep learning, and discussed earlier in Section 5.5. To recap: SWAG exploits the fact that SGD itself can be viewed as approximate Bayesian inference (Mandt et al., 2017),

and uses an ensemble of SGD iterates saved during the later stages of training to fit an approximate low-rank Gaussian distribution over the weights of the DNN. During inference, samples from this distribution are used to create an ensemble of models; predictions are the average of this ensemble. We use an ensemble size of 30 as proposed in the original paper.

Datasets

To allow experimenting with multiple algorithms and tuning of hyper-parameters, while at the same time maintaining a challenging level of task difficulty, we choose the Tiny ImageNet (tin, 2017) as our in-distribution dataset, which consists of 200 categories of objects, with 500 train and 50 validation images per category. We split the set into two folds of 100 classes each, training on the first fold, and using the second fold as a challending OoD dataset. We also use the following additional datasets to test detection performance.

- **ImageNet** We use a 10-class subset of ImageNet (Deng et al., 2009), where the classes do not overlap with those in the Tiny ImageNet set.

- **MIT Places** (Zhou et al., 2017)A dataset for scene recognition containing images from more than 400 scene categories.

- **Gaussian Noise** A synthetically generated dataset consisting of random Gaussian noise images, where each pixel is sampled from an i.i.d Gaussian distribution.

- **Tiny ImageNet cross-folds** This is an *intra-dataset* task, where we train on one fold, and test OoD detection on other folds. Even though the folds have non-overlapping classes, there are *similar* classes across folds; hence this is the most challenging test in our experiments.

Both the ID and OoD datasets were scaled to $96 \times 96 \times 3$ RGB format. For the OoD dataset used for training, we use a large corpus of unlabeled data from the STL-10 (Coates

et al., 2011) dataset, consisting of 100K $96 \times 96 \times 3$ RGB images; for completeness, we also test the detection performance on a non-overlapping portion of the same dataset.

Setup

For all our experiments in this section, we use the VGG-16 (Simonyan and Zisserman, 2014) architecture, training for 200 epochs using SGD with Nesterov momentum, annealing the learning rate at epoch $60, 120$ and 160.

Metrics

As before, we use the AUROC and FPR-95 metrics to report performance, where the in- and out-of-distribution datas are treated as the negative and positive classes respectively.

Results

Detailed results are shown in Table 7.3, where the best performing methods for each metric are shown in bold. The DAC is the only method in this set of experiments that uses an augmented dataset, and as is clearly evident from the results, this bestows a huge advantage over the other methods in most cases.

On the ImageNet, Places-500 and the STL-10 datasets, the DAC has near-perfect performance, which likely indicates that these OoD test sets have similar distribution to the OoD dataset used in training (unlabeled STL-10 images)[2] Nevertheless, we know from the experiments in the previous section that even in the case where the OoD test set is significantly different from the OoD set used to augment training, the presence of the abstention class still confers significant performance advantages to the DAC.

The high variance of the other uncertainty methods on Gaussian noise, and the equally conspicuous consistency of the DAC is especially remarkable. We have discussed pathological behavior on Gaussian noise in previous chapters, and the experiments here confirm that

[2]This is of course true for the STL-10 dataset used while testing. While this set also includes unseen classes, we nevertheless report the peformance here for completeness.

Method \ $\mathcal{D}_{out}$	$\mathcal{D}_{in}$: Tiny ImageNet (Fold 0) (Model: VGG-16)				
	ImageNet	**Places-500**	**Gaussian**	**STL-10**	**Fold 1**
	AUROC ↑				
Softmax	$73.20_{0.48}$	$75.85_{0.73}$	$50.27_{10.95}$	$71.54_{0.32}$	$73.65_{0.36}$
Entropy	$73.99_{0.55}$	$76.76_{0.68}$	$51.72_{12.17}$	$72.00_{0.34}$	$74.43_{0.37}$
MC Dropout	$72.38_{0.50}$	$75.89_{0.67}$	$52.32_{11.11}$	$71.12_{0.29}$	$73.83_{0.36}$
Temp Scaling	$82.35_{10.01}$	$85.39_{12.06}$	$64.60_{21.54}$	$73.94_{9.28}$	$72.19_{4.51}$
Mixup	$73.37_{0.66}$	$76.98_{1.05}$	$98.21_{6.12}$	$73.80_{0.34}$	$\mathbf{76.35}_{0.53}$
Deep Ensemble	$72.55_{0.36}$	$74.40_{0.79}$	$66.04_{15.61}$	$69.93_{0.34}$	$72.96_{0.36}$
SWAG	$71.83_{0.95}$	$72.58_{0.59}$	$57.39_{12.10}$	$69.23_{0.96}$	$71.54_{0.57}$
DAC (Ours)	$\mathbf{99.29}_{0.45}$	$\mathbf{100.00}_{0.00}$	$\mathbf{100.00}_{0.00}$	$\mathbf{100.00}_{0.00}$	$73.41_{0.46}$
	FPR95 ↓				
Softmax	$64.26_{0.48}$	$59.15_{0.73}$	$58.35_{10.95}$	$69.17_{0.32}$	$62.04_{0.36}$
Entropy	$64.29_{0.55}$	$59.12_{0.68}$	$57.70_{12.17}$	$69.28_{0.34}$	$61.92_{0.37}$
MC Dropout	$67.70_{0.50}$	$60.18_{0.67}$	$57.06_{11.11}$	$70.73_{0.29}$	$62.86_{0.36}$
Temp Scaling	$48.28_{10.01}$	$36.48_{12.06}$	$43.85_{21.54}$	$67.57_{9.28}$	$65.52_{4.51}$
Mixup	$66.22_{0.66}$	$58.49_{1.05}$	$4.91_{6.12}$	$66.94_{0.34}$	$\mathbf{58.84}_{0.53}$
Deep Ensemble	$66.82_{0.36}$	$62.48_{0.79}$	$43.03_{15.61}$	$71.70_{0.34}$	$63.78_{0.36}$
SWAG	$65.84_{1.00}$	$68.80_{1.00}$	$62.20_{10.33}$	$72.68_{0.99}$	$66.24_{0.94}$
DAC (Ours)	$\mathbf{1.33}_{0.45}$	$\mathbf{0.00}_{0.00}$	$\mathbf{0.00}_{0.00}$	$\mathbf{0.00}_{0.00}$	$64.91_{0.46}$

Table 7.3: OoD detection performance of DAC compared to approaches that model uncertainty in deep learning. The performance of DAC as an OoD detector, evaluated on various metrics and compared against competing baselines. ↑ and ↓ indicate that higher and lower values are better, respectively. Best performing methods (ignoring statistically insignificant differences)on each metric are in bold. All methods were re-implemented using open-sourced code where available.

DNNs are indeed very susceptible to overconfident predictions on random noise data. The only other method that achieves good performance in this regard is Mixup, which is consistent with our previous findings (see Chapter 6). Calibration methods like temperature scaling, while producing well calibrated scores on in-distribution data, end up reducing the confidence on in-distribution data as well, and thus losing discriminative power between the two types of data; this is reflected in the poor performance of temperature scaling as a method for OoD

detection.

The intra-dataset (i.e., cross-fold test, shown in the last column) is the only test where the abstention approach does not seem to work well (in an absolute sense, even though it is better than quite a few of the other methods). No method performs reliably in this test, and this is a reflection of the fact that the classes in Tiny ImageNet are remarkably similar to each other, and it is thus quite difficult to distinguish intra-dataset classes reliably; in fact, the classification accuracy on the full set for state-of-art methods is less than 70% (tin, 2017).

Finally we also note here that many of the methods listed in the table, like temperature scaling and deep ensembles, can be combined with the abstention approach. Indeed, the addition of an extra abstention class and training with OoD data is compatible with most uncertainty modeling techniques in deep learning; we leave the exploration of such combination approaches for future work.

7.5 Conclusion

We presented *a simple, but highly effective* method for open set and out-of-distribution detection that clearly demonstrated the efficacy of using an extra abstention class and augmenting the training set with outliers. While previous work has shown the efficacy of training with outliers (Hendrycks et al., 2018a), here we demonstrated a simpler approach for exploiting outlier data that further improves upon existing methods, while also being significantly simpler. The simplicity of implementation, absence of additional hyperparameter tuning and computational efficiency during testing makes this a very viable approach for improving out-of-distribution and novel category detection in real-world deployments. We hope that this will also serve as an effective baseline for comparing future work in this domain.

Chapter 8

FUTURE DIRECTIONS

The work in this thesis was motivated by practical challenges that are encountered when using deep learning models for classification, both during training and prediction. In Part I, using a novel, abstention-based framework, we were able to show how DNNs can be robustly trained in the presence of noisy data, and in addition, also *learn features* that are indicative of systematic corruption in the data; further improvements were demonstrated by applying PID control and semi-supervised learning to the problem of learning in the presence of label noise.

The work in Part I can be extended in a number of ways. While our focus in this thesis was on classification problems, the abstention formulation can also be extended to *regression* tasks where the model needs to predict a real, continuous value. One possibility here is to replace the cross-entropy loss with the squared error loss, while at the same time retaining the extra abstention class and the associated abstention penalty. A mixed classification-regression formulation for object detection was discussed in (Kendall and Gal, 2017); a similar approach could be adapted to the abstention formulation as well.

We can also extend the classification framework to other tasks; for example, label noise can also occur in object detection (Szegedy et al., 2013a) and semantic segmentation tasks (Long et al., 2015), where the ground truth labels are now at the pixel or super-pixel level. Here, the DAC loss function would need to be reformulated at a finer level of granularity; the advantage here is that the DAC loss is architecture independent, and theoretically, should work even with segmentation networks.

A very active sub-field of generative modeling in deep learning is Generative Adversarial Networks (Goodfellow et al., 2014a), that have shown to be capable of generating highly

realistic images (Karras et al., 2019). GANs, however, are notoriously hard to train and stabilize (Salimans et al., 2016), so an interesting possibility here is to use the DAC as the discriminative component in the GAN. The idea here is that the GAN discriminator would be initially allowed to abstain on a large number of generated samples, but as the discriminator gets better, abstention is decayed.

The abstention approach was only tested in benign settings in this thesis. However, the principle of abstention can be a powerful defense against adversarial attacks, where the model is allowed to abstain instead of making a prediction. In general, it is hard to defend against such attacks (Papernot, 2018), but under some reasonable assumptions – where the adversary might not have entire knowledge of the model – one could potentially combine adversarial detection with abstention for a viable defense against such attacks.

Another area of security in machine learning deals with robustness against so-called *data poisoning attacks* (Chen et al., 2017), where the training labels are *maliciously corrupted* such that the DNN makes a targeted misclassification during test time. For example, there might be pictures of faces of unauthorized persons in the training data of a face detection system that have been deliberately mapped to labels of authorized faces. The main challenge here is that data poisoning is mathematically not a form of label noise; the mapping from multiple images of an unauthorized user to the identifier (label) of an authorized user, while malicious, is not inconsistent. However, we can exploit the fact that poisoned samples are outliers in *feature space*, and instead of cross-entropy, we could potentially use a loss function that penalizes the feature-space dissimilarity of a sample with its neighbors that have the same label. Setting the abstention setpoint to a reasonably small fraction of the training data – by using the PID control technique we developed in Chapter 4 – might afford some degree of protection against such attacks.

The exploration of the causes of uncertainty and miscalibration in deep learning that we undertook in Part 2 allowed us to improve threshold-based abstention techniques; using this, and further leveraging our results from Part I, we were also able to demonstrate an effective method for open set and out-of-distribution detection for image classification. There are a

number of interesting directions that could be pursued here as well: our current experimental setting require the usage of additional OoD samples for training the abstention class. An alternative approach here would be to use GANs for generating synthetic samples that can then be used as OoD data, similar to the work in (Ge et al., 2017), (Neal et al., 2018).

Also, our experiments on open set detection were restricted to images, but a similar approach could also be applied to acoustic and other types of signals. With AI computation increasingly occurring on edge devices (Verhelst and Moons, 2017) like mobile phones and sensors – the so called Internet-of-Things (Atzori et al., 2010) – and the increasing presence of deep learning classification systems in this domain (Li et al., 2018), a robust classification system that can reliably detect signals of interest and reject false positives would have great value in real-world applications. Indeed, a very worthwhile area of further exploration would be to investigate how predictive uncertainty and miscalibration are affected for deep learning models that have been sparsified and compressed (Iandola et al., 2016; Lane et al., 2016) for deployment on edge devices. In such cases, a computationally efficient means of capturing uncertainty, like the one we developed in Chapter 7, would be especially useful.

Since the trends of increasing compute power and large data are expected to continue well into the future, one can also assume that deep learning-based AI will play an increasingly significant role in technology and society; it is hoped that the work in this thesis has provided practical insights into training and deploying these systems in a safe and robust manner.

BIBLIOGRAPHY

(2017). Tiny imagenet visual recognition challenge.

Akhtar, N. and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430.

Alexandrescu, A. and Kirchhoff, K. (2009). Graph-based learning for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 119–127. Association for Computational Linguistics.

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.

An, W., Wang, H., Sun, Q., Xu, J., Dai, Q., and Zhang, L. (2018). A pid controller approach for stochastic optimization of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8522–8531.

Ang, K. H., Chong, G., and Li, Y. (2005). Pid control system analysis, design, and technology. *IEEE transactions on control systems technology*, 13(4):559–576.

Angelova, A., Abu-Mostafam, Y., and Perona, P. (2005). Pruning training sets for learning of object categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 494–501. IEEE.

Angkititrakul, P. and Hansen, J. H. (2004). Identifying in-set and out-of-set speakers using neighborhood information. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–393. IEEE.

Angluin, D. and Laird, P. (1988). Learning from noisy examples. *Machine Learning*, 2(4):343–370.

Arazo, E., Ortego, D., Albert, P., O'Connor, N. E., and McGuinness, K. (2019). Unsupervised label noise modeling and loss correction. *arXiv preprint arXiv:1904.11238*.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. (2016). Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*.

Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. (2017). A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*.

Asuncion, A. and Newman, D. (2007). Uci machine learning repository.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15):2787–2805.

Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156.

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482.

Bartlett, P. L. and Wegkamp, M. H. (2008). Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(Aug):1823–1840.

Bendale, A. and Boult, T. E. (2016). Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Bennett, S. (1993). Development of the pid controller. *IEEE Control Systems Magazine*, 13(6):58–62.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. (2019). Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Boucheron, S., Lugosi, G., and Bousquet, O. (2003). Concentration inequalities. In *Summer School on Machine Learning*, pages 208–240. Springer.

Boult, T., Cruz, S., Dhamija, A., Gunther, M., Henrydoss, J., and Scheirer, W. (2019). Learning and the unknown: Surveying steps toward open world recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9801–9807.

Brodley, C. E. and Friedl, M. A. (1999). Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167.

Bui, T. D., Ravi, S., and Ramavajjala, V. (2017). Neural graph machines: Learning neural networks using graphs. *arXiv preprint arXiv:1703.04818*.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Chalkidis, I. and Kampas, D. (2019). Deep learning in law: early adaptation and legal word embeddings trained on large corpora. *Artificial Intelligence and Law*, 27(2):171–198.

Chapelle, O., Schölkopf, B., Zien, A., et al. (2006). *Semi-supervised learning*. MIT press Cambridge.

Chapelle, O., Weston, J., Bottou, L., and Vapnik, V. (2001). Vicinal risk minimization. In *Advances in neural information processing systems*, pages 416–422.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. (2017). Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Chow, C. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46.

Chow, C.-K. (1957). An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, (4):247–254.

Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223.

Colton, D., Fanty, M., and Cole, R. A. (1995). Utterance verification improves closed-set recognition and out-of-vocabulary rejection. In *Fourth European Conference on Speech Communication and Technology*.

Cordella, L. P., De Stefano, C., Tortorella, F., and Vento, M. (1995). A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 6(5):1140–1147.

Cortes, C., DeSalvo, G., and Mohri, M. (2016). Learning with rejection. In *International Conference on Algorithmic Learning Theory*, pages 67–82. Springer.

Crammer, K. and Globerson, A. (2012). Discriminative learning via semidefinite probabilistic models. *arXiv preprint arXiv:1206.6815*.

De Stefano, C., Sansone, C., and Vento, M. (2000). To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(1):84–94.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee.

Deng, J. and Hu, Q. (2003). Open set text-independent speaker recognition based on set-score pattern classification. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 2, pages II–73. IEEE.

DeVries, T. and Taylor, G. W. (2018). Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*.

Dhamija, A. R., Günther, M., and Boult, T. (2018). Reducing network agnostophobia. In *Advances in Neural Information Processing Systems*, pages 9157–9168.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157.

Ding, Y., Wang, L., Fan, D., and Gong, B. (2018). A semi-supervised two-stage approach

to learning from noisy labels. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1215–1224. IEEE.

Falcini, F., Lami, G., and Costanza, A. M. (2017). Deep learning in automotive software. *IEEE Software*, 34(3):56–63.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

Fei-Fei, L. and Deng, J. (2017). Imagenet: Where have we been; where are we going?

Frénay, B. and Verleysen, M. (2014). Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869.

Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Fumera, G. and Roli, F. (2002). Support vector machines with embedded reject option. In *International Workshop on Support Vector Machines*, pages 68–82. Springer.

Gal, Y. (2016). Uncertainty in deep learning. *University of Cambridge*, 1:3.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Ge, Z., Demyanov, S., Chen, Z., and Garnavi, R. (2017). Generative openmax for multi-class open set classification. *arXiv preprint arXiv:1707.07418*.

Geifman, Y. and El-Yaniv, R. (2017). Selective classification for deep neural networks. In *Advances in neural information processing systems*, pages 4885–4894.

Geifman, Y. and El-Yaniv, R. (2019). Selectivenet: A deep neural network with an integrated reject option. *arXiv preprint arXiv:1901.09192*.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58.

Ghosh, A., Kumar, H., and Sastry, P. (2017). Robust loss functions under label noise for deep neural networks. In *AAAI*, pages 1919–1925.

Goldberger, J. and Ben-Reuven, E. (2016). Training deep neural-networks using a noise adaptation layer.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Görnitz, N., Porbadnigk, A., Binder, A., Sannelli, C., Braun, M., Müller, K.-R., and Kloft, M. (2014). Learning and evaluation in presence of non-iid label noise. In *Artificial Intelligence and Statistics*, pages 293–302.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Grandvalet, Y. and Bengio, Y. (2005). Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536.

Grandvalet, Y., Rakotomamonjy, A., Keshet, J., and Canu, S. (2009). Support vector machines with a reject option. In *Advances in neural information processing systems*, pages 537–544.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*.

Guo, H., Mao, Y., and Zhang, R. (2018). Mixup as locally linear out-of-manifold regularization. *arXiv preprint arXiv:1809.02499*.

Guo, H., Mao, Y., and Zhang, R. (2019). Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. (2018). Co-teaching: robust training deep neural networks with extremely noisy labels. *arXiv preprint arXiv:1804.06872*.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hein, M., Andriushchenko, M., and Bitterwolf, J. (2018). Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *arXiv preprint arXiv:1812.05720*.

Hellman, M. E. (1970). The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):179–185.

Hendrycks, D. and Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

Hendrycks, D., Mazeika, M., and Dietterich, T. (2018a). Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*.

Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K. (2018b). Using trusted data to train deep networks on labels corrupted by severe noise. *arXiv preprint arXiv:1802.05300*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hosseini, H. (2019). *Machine Learning in Adversarial Settings: Attacks and Defenses*. PhD thesis.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Hu, W., Li, Z., and Yu, D. (2019). Understanding generalization of deep neural networks trained with noisy labels. *arXiv preprint arXiv:1905.11368*.

Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., et al. (2015). An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Inoue, H. (2018). Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*.

Jiang, H., Kim, B., Guan, M., and Gupta, M. (2018a). To trust or not to trust a classifier. In *Advances in neural information processing systems*, pages 5541–5552.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. (2018b). Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2309–2318.

Júnior, P. R. M., De Souza, R. M., Werneck, R. d. O., Stein, B. V., Pazinato, D. V., de Almeida, W. R., Penatti, O. A., Torres, R. d. S., and Rocha, A. (2017). Nearest neighbors distance ratio open-set classifier. *Machine Learning*, 106(3):359–386.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Laine, S. and Aila, T. (2016). Temporal ensembling for semi-supervised learning.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413.

Lane, N. D., Bhattacharya, S., Georgiev, P., Forlivesi, C., Jiao, L., Qendro, L., and Kawsar, F. (2016). Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12. IEEE.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2.

Lee, K., Lee, H., Lee, K., and Shin, J. (2017). Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*.

Lee, K., Lee, K., Lee, H., and Shin, J. (2018). A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177.

Leibig, C., Allken, V., Ayhan, M. S., Berens, P., and Wahl, S. (2017). Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):17816.

Lerman, K., Blair-Goldensohn, S., and McDonald, R. (2009). Sentiment summarization: evaluating and learning user preferences. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 514–522. Association for Computational Linguistics.

Lewis-Kraus, G. (2016). The great ai awakening. *The New York Times Magazine*, 14:12.

Li, F. and Wechsler, H. (2005). Open set face recognition using transduction. *IEEE transactions on pattern analysis and machine intelligence*, 27(11):1686–1697.

Li, H., Ota, K., and Dong, M. (2018). Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE network*, 32(1):96–101.

Li, M., Soltanolkotabi, M., and Oymak, S. (2019). Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. *arXiv preprint arXiv:1903.11680*.

Li, W., Wang, L., Li, W., Agustsson, E., and Van Gool, L. (2017a). Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*.

Li, X. and Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., and Li, L.-J. (2017b). Learning from noisy labels with distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1910–1918.

Liang, D., Yang, F., Zhang, T., and Yang, P. (2018a). Understanding mixup training methods. *IEEE Access*, 6:58774–58783.

Liang, S., Li, Y., and Srikant, R. (2018b). Enhancing the reliability of out-of-distribution image detection in neural networks.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88.

Liu, T. and Tao, D. (2015). Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461.

Liu, Y. and Kirchhoff, K. (2013). Graph-based semi-supervised learning for phone and segment classification. In *INTERSPEECH*, pages 1840–1843.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Long, P. M. and Servedio, R. A. (2010). Random classification noise defeats all convex potential boosters. *Machine learning*, 78(3):287–304.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2019). A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13132–13143.

Malach, E. and Shalev-Shwartz, S. (2017). Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, pages 960–970.

Malkin, J. and Bilmes, J. (2008). Ratio semi-definite classifiers. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4113–4116. IEEE.

Malkin, J. and Bilmes, J. (2009). Multi-layer ratio semi-definite classifiers. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4465–4468. IEEE.

Malkin, J., Subramanya, A., and Bilmes, J. A. (2009). On the semi-supervised learning of multi-layered perceptrons. In *INTERSPEECH*, pages 660–663.

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907.

Marcus, G. and Davis, E. (2019). *Rebooting AI: building artificial intelligence we can trust*. Pantheon.

McDonald, R. A., Hand, D. J., and Eckley, I. A. (2003). An empirical comparison of three boosting algorithms on real data sets with artificial class noise. In *International Workshop on Multiple Classifier Systems*, pages 35–44. Springer.

Melville, P., Shah, N., Mihalkova, L., and Mooney, R. J. (2004). Experiments on ensembles with missing and noisy data. In *International Workshop on Multiple Classifier Systems*, pages 293–302. Springer.

Miotto, R., Li, L., Kidd, B. A., and Dudley, J. T. (2016). Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6:26094.

Mitros, J. and Mac Namee, B. (2019). On the validity of bayesian neural networks for uncertainty estimation. *arXiv preprint arXiv:1912.01530*.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nam, J. G., Park, S., Hwang, E. J., Lee, J. H., Jin, K.-N., Lim, K. Y., Vu, T. H., Sohn, J. H., Hwang, S., Goo, J. M., et al. (2019). Development and validation of deep learning–based automatic detection algorithm for malignant pulmonary nodules on chest radiographs. *Radiology*, 290(1):218–228.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. (2013). Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204.

Neal, L., Olson, M., Fern, X., Wong, W.-K., and Li, F. (2018). Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–628.

Neal, R. M. (1995). *BAYESIAN LEARNING FOR NEURAL NETWORKS*. PhD thesis, University of Toronto.

Nettleton, D. F., Orriols-Puig, A., and Fornells, A. (2010). A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33(4):275–306.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436.

Okamoto, S. and Yugami, N. (1997). An average-case analysis of the k-nearest neighbor classifier for noisy domains. In *IJCAI (1)*, pages 238–245.

Osawa, K., Swaroop, S., Khan, M. E. E., Jain, A., Eschenhagen, R., Turner, R. E., and Yokota, R. (2019). Practical deep learning with bayesian principles. In *Advances in Neural Information Processing Systems*, pages 4289–4301.

Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Papernot, N. (2018). Characterizing the limits and defenses of machine learning in adversarial settings.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

Patrini, G., Rozza, A., Menon, A. K., Nock, R., and Qu, L. (2017). Making deep neural networks robust to label noise: A loss correction approach. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, pages 2233–2241.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Pietraszek, T. (2005). Optimizing abstaining classifiers using roc analysis. In *Proceedings of the 22nd international conference on Machine learning*, pages 665–672.

Porbadnigk, A. K., Görnitz, N., Sannelli, C., Binder, A., Braun, M., Kloft, M., and Müller, K.-R. (2014). When brain and behavior disagree: Tackling systematic label noise in eeg data with machine learning. In *Brain-Computer Interface (BCI), 2014 International Winter Workshop on*, pages 1–4. IEEE.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Rashtchian, C., Young, P., Hodosh, M., and Hockenmaier, J. (2010). Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147. Association for Computational Linguistics.

Rasmussen, C. E. and Nickisch, H. (2010). Gaussian processes for machine learning (gpml) toolbox. *Journal of machine learning research*, 11(Nov):3011–3015.

Recht, B. (2018). The best things in life are model free.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. (2014). Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2018). Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*.

Rocco, P. (1996). Stability of pid control for industrial robot arms. *IEEE transactions on robotics and automation*, 12(4):606–614.

Sajjadi, M., Javanmardi, M., and Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171.

Sakia, R. (1992). The box-cox transformation technique: a review. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(2):169–178.

Salih, A. L., Moghavvemi, M., Mohamed, H. A., and Gaeid, K. S. (2010). Modelling and pid controller design for a quadrotor unmanned air vehicle. In *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, volume 1, pages 1–5. IEEE.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242.

Scheirer, W. J., de Rezende Rocha, A., Sapkota, A., and Boult, T. E. (2012). Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772.

Scheirer, W. J., Jain, L. P., and Boult, T. E. (2014). Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2317–2324.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., et al. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626.

Sensoy, M., Kaplan, L., and Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems*, pages 3179–3189.

Shen, Y. and Sanghavi, S. (2019). Learning with bad training data via iterative trimmed loss minimization. In *International Conference on Machine Learning*, pages 5739–5748.

Shu, L., Xu, H., and Liu, B. (2017). Doc: Deep open classification of text documents. *arXiv preprint arXiv:1709.08716*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, H., Kim, M., and Lee, J.-G. (2019). Selfie: Refurbishing unclean samples for robust deep learning. In *International Conference on Machine Learning*, pages 5907–5915.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Stallkamp, J., Ekenel, H. K., and Stiefelhagen, R. (2007). Video-based face recognition on real-world data. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.

Subramanya, A. and Bilmes, J. (2008). Soft-supervised learning for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1090–1099. Association for Computational Linguistics.

Subramanya, A., Petrov, S., and Pereira, F. (2010). Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176. Association for Computational Linguistics.

Subramanya, A., Srinivas, S., and Babu, R. V. (2017). Confidence estimation in deep neural networks via density modelling. *arXiv preprint arXiv:1707.07013*.

Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., and Fergus, R. (2014). Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*.

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 2. MIT press Cambridge.

Szegedy, C., Toshev, A., and Erhan, D. (2013a). Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013b). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tanaka, D., Ikami, D., Yamasaki, T., and Aizawa, K. (2018). Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560.

Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204.

Thulasidasan, S., Bhattacharya, T., Bilmes, J., Chennupati, G., and Mohd-Yusof, J. (2019a). Combating label noise in deep learning using abstention. In *International Conference on Machine Learning*, pages 6234–6243.

Thulasidasan, S. and Bilmes, J. (2017). Acoustic classification using semi-supervised deep neural networks and stochastic entropy-regularization over nearest-neighbor graphs. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2731–2735. IEEE.

Thulasidasan, S., Bilmes, J., and Kenyon, G. (2016). Efficient distributed semi-supervised learning using stochastic regularization over affinity graphs. *arXiv preprint arXiv:1612.04898*.

Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T., and Michalak, S. (2019b). On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 13888–13899.

Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970.

Vahdat, A. (2017). Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5596–5605.

Valiant, L. G. (1984). A theory of the learnable. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445. ACM.

Vanschoren, J. (2018). Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*.

Vapnik, V. N. and Chervonenkis, A. Y. (2015). On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer.

Veit, A., Alldrin, N., Chechik, G., Krasin, I., Gupta, A., and Belongie, S. (2017). Learning from noisy large-scale datasets with minimal supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 839–847.

Verhelst, M. and Moons, B. (2017). Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to iot and edge devices. *IEEE Solid-State Circuits Magazine*, 9(4):55–65.

Verma, V., Lamb, A., Beckham, C., Courville, A., Mitliagkis, I., and Bengio, Y. (2018). Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *arXiv preprint arXiv:1806.05236*.

Vernekar, S., Gaurav, A., Abdelzad, V., Denouden, T., Salay, R., and Czarnecki, K. (2019). Out-of-distribution detection in classifiers via generation. *arXiv preprint arXiv:1910.04241*.

Vyas, A., Jammalamadaka, N., Zhu, X., Das, D., Kaul, B., and Willke, T. L. (2018). Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 550–564.

Wang, H. and Yeung, D.-Y. (2016). Towards bayesian deep learning: A framework and some

existing methods. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3395–3408.

Weston, J., Ratle, F., and Collobert, R. (2008). Deep learning via semi-supervised embedding. In *Proceedings of the 25th international conference on Machine learning*, pages 1168–1175.

Wilson, A. G. (2020). The case for bayesian deep learning. *arXiv preprint arXiv:2001.10995*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. (2015). Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Yan, S. and Wang, H. (2009). Semi-supervised learning by sparse representation. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 792–801. SIAM.

Yoshihashi, R., Shao, W., Kawakami, R., You, S., Iida, M., and Naemura, T. (2019). Classification-reconstruction learning for open-set recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4016–4025.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

Yu, X., Han, B., Yao, J., Niu, G., Tsang, I., and Sugiyama, M. (2019). How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pages 7164–7173.

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., et al. (2013). On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhang, H. https://github.com/hongyi-zhang/mixup.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, Z. and Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*.

Zhao, P., Chen, J., Song, Y., Tao, X., Xu, T., and Mei, T. (2012). Design of a control system for an autonomous vehicle based on adaptive-pid. *International Journal of Advanced Robotic Systems*, 9(2):44.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2017). Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., and Xu, B. (2016). Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.

Zhou, T. and Bilmes, J. A. (2018). Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *ICLR*.

Zhu, X., Lafferty, J., and Rosenfeld, R. (2005). *Semi-supervised learning with graphs*. Carnegie Mellon University, Language Technologies Institute, School of Computer Science.

Zhu, X. and Wu, X. (2004). Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22(3):177–210.

Zhu, X., Wu, X., and Chen, Q. (2003). Eliminating class noise in large datasets. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 920–927.

Ziegler, J. G. and Nichols, N. B. (1942). Optimum settings for automatic controllers. *trans. ASME*, 64(11).