Building the Next Generation of Multimodal Models

Gabriel Ilharco

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

Doctor of Philosophy

University of Washington

2024

Reading Committee:

Hannaneh Hajishirzi, Chair

Ali Farhadi, Chair

Luke Zettlemoyer

Zaid Harchaoui

Program Authorized to Offer Degree:
Paul G. Allen School of Computer Science and Engineering

University of Washington

Abstract

Building the Next Generation of Multimodal Models

Gabriel Ilharco

Co-Chairs of the Supervisory Committee:
Hannaneh Hajishirzi
Paul G. Allen School of Computer Science and Engineering

Ali Farhadi
Paul G. Allen School of Computer Science and Engineering

One of the fundamental goals of machine learning is to create systems capable of processing data from a variety of modalities such as images and text. I argue that the next generation of multimodal models will be enabled by a deeper understanding of how to design pretraining datasets, and by techniques that offer better control over models after pretraining. Towards the first goal, I introduce a fully open-source benchmark for designing multimodal datasets. This benchmark provides a shared experimental setting for research on dataset curation, allowing researchers to conduct rigorous and controlled experiments. Our experiments highlight the potential of rigorous empirical work on dataset curation, finding pretraining datasets that outperform existing datasets by a large margin. Towards the second goal, I present multiple techniques for improving models after pretraining. Our fine-tuning techniques improve accuracy without overspecialization and without increasing inference costs. Moreover, I present a modular framework for steering the behavior of trained models, designed to efficiently add or delete capabilities while operating directly within the models' weight space. Altogether, these new techniques pave the way for the next generation of multimodal models.

TABLE OF CONTENTS

DEDICATION

To my advisors, Hanna and Ali. Thank you for believing in me and giving me a chance when many would not. Thank you for creating an environment where I could learn and grow beyond my wildest expectations. Thank you for being so supportive in my academic journey, for your patience while fleshing out eccentric ideas, for all your insights when designing and running experiments, and for bearing with my stubbornness throughout the years. Thank you for making me a better researcher, student, and person. Being advised by you has been one of the most meaningful experiences of my life, and one I will cherish for the rest of it.

To my mentors. To Ludwig, thank you for inspiring me to never be afraid of tackling ambitious, hard problems. Thank you for encouraging me to think long-term, and for showing me the joys of working with large groups of people on challenging projects. To Marco, thank you for teaching me how to think clearly, from first principles, about what really matters in research and in life. To Jason, who introduced me to research in a way that made me fall in love with it. I am very grateful for all the lessons and support throughout the years.

To my dear family. To my parents José e Filomena, who always supported me in every possible way. Thank you for the love and all the sacrifices you have made to make my life better. To my brother César and my sister Abby, who were there in every step along the way. Thank you for your love and constant presence, and thank you for inspiring me to always be the best version of myself. The physical distance that tends to separate us only deepens my appreciation for the bond we share. I am profoundly grateful for the example you set throughout the years, which shaped the person I am today.

To my close friends, Alex, Ana, Amanda, Archit, Ari, Isa, Ivan, Jemma, Lara, Marcio, Marco, Matheus, Mitchell, Ofir, Stephanie, Thauan, Tim, and Vini. You were a light in my life, and through your presence, made every step of the way better. For all the laughs,

songs, stories and company, thank you. To all friends Aditya, Ana, Andre, Bruna, Eduardo, Felipe F., Felipe R., Guilherme, Helena, Lucas, Marina, Martin, Matt, Melanie, Nish, Paulo, Sam, Samir, Sarah, Suchin, Thiago O., Thiago P., Victor, Vivek, and others who I might be missing but appreciate nonetheless. I cherish you deeply, and hope we can continue in each others lives for many years to come.

To my labmates Aaron, Aditya, Ainaz, Akari, Alex, Bhargavi, Colin, Daniel, Dave, Dhruba, Ellen, Gary, Hamish, James, Jongwoon, Keivan, Kiana, Kuo-Hao, Matt D., Matt W., Mitchell, Qingqing, Reza, Rik, Rowan, Sachin, Sarah, Sewon, Thao, Vivek, Yanai, and Yizhong. Thank you for all the ideas we shared, experiments we discussed and for all we were able to learn together. I sincerely hope our paths cross again in the near future.

To all my collaborators Achal, Akari, Alex, Ali, Ally, Alon, Amnon, Ana, Ananth, Anas, Ari, Ashish, Becca, Ben, Cade, Christoph, Dan, Dani, Daniel, Dheeru, Dhruba, Dirk, Eric, Eugene, Eyal, Georgios, Hanna, Hao, Haoshuo, Harsh, Hong, Ian, Irena, Jack, Jason, Jenia, Jesse, Jiagming, Jieyu, Jonathan, Jong Wook, Joongwon, Josh, Jungo, Kalyani, Kevin, Kiana, Klemen, Maarten, Marco, Margaret, Matt, Mehdi, Mitchell, Nelson, Nicholas, Nikolaos, Nitish, Noah, Olga, Ori, Pang Wei, Peter, Phoebe, Pradeep, Qiang, Ranjay, Rapha, Reut, Rohan, Romain, Roozbeh, Ross, Rowan, Roy, Ryan, Sameer, Samir, Sanjay, Scott, Sewon, Sewoong, Sihao, Simon, Stephen, Suchin, Thao, Vaishaal, Victoria, Vihan, Wanrong, Weizhu, William, Yair, Yanai, Yi, Yizhe, Yizhong, Yoav, Yonatan, Yuhao, and Yusuf. This thesis is really your work, without you none of it would be possible. Collaborating with you has been a rich and rewarding experience, and I am deeply thankful for having been able to built more together with you. Thank you.

Chapter 1

INTRODUCTION

Recent advances in multimodal learning such as CLIP (Radford et al., 2021), DALL-E (Ramesh et al., 2021, 2022), Stable Diffusion (Rombach et al., 2022), Flamingo (Alayrac et al., 2022), and GPT-4 (OpenAI, 2023) offer unprecedented generalization capabilities in zero-shot classification, image generation, and in-context learning. While these advances use different algorithmic techniques, e.g., contrastive learning, diffusion, or auto-regressive modeling, they all rest on a common foundation: large datasets containing paired image-text examples. For instance, CLIP's training set contains 400 million image-text pairs, and Stable Diffusion was trained on the two billion examples from LAION-2B (Schuhmann et al., 2022), a dataset of more than two billion image-text pairs. This new generation of image-text datasets is 1,000 times larger than previous datasets such as ImageNet, which contains 1.2M images (Deng et al., 2009; Russakovsky et al., 2015).

Despite the central role of datasets (Fang et al., 2022), little is known about them. Many state-of-the-art datasets are proprietary, and even for public datasets such as LAION-2B (Schuhmann et al., 2022), it is unclear how design choices such as the data source or filtering techniques affect the resulting models. While there are thousands of ablation studies for algorithmic design choices (loss function, model architecture, etc.), datasets are often treated as monolithic artifacts without detailed investigation. Moreover, datasets currently lack the benchmark-driven development process that has enabled a steady stream of improvements on the model side and isolates data enhancements from changes to the model. These issues impede further progress in multimodal learning, as evidenced by recent work showing that public datasets currently do not match the scaling behavior of proprietary alternatives (Cherti et al., 2022).

In the first part of this disseration, we focus on showing the importance of pretraining datasets (Fang et al., 2022), and taking a step towards a more rigorous dataset development

process by introducing **DataComp, a new benchmark for multimodal dataset design** (Gadre et al., 2023). DataComp flips the traditional benchmarking paradigm in machine learning where the dataset is fixed and researchers propose new training algorithms. Instead, we hold the entire training code and computational budget constant so that participants innovate by proposing new training sets.

DataComp focuses on two key challenges that arise when assembling large training datasets: what data sources to train on, and how to filter a given data source. Each challenge corresponds to one track in our benchmark. To facilitate the *filtering track*, we introduce CommonPool, a dataset of 12.8B image-text pairs collected from Common Crawl and currently the largest public image-text dataset. In this track, the goal of participants is to find the best subset of CommonPool to train on. In the second track, participants may leverage any data source, as long as it does not overlap with our evaluation testbed.

We present over three hundred baseline experiments on dataset curation, including techniques such as querying captions for relevant keywords, filtering based on image embeddings, and applying a threshold on CLIP scores. A key result from our baselines experiments is that smaller, more stringently filtered datasets can lead to models that generalize *better* than larger datasets coming from the same pool. Our best filtering baseline increases ImageNet zero-shot accuracy by 6.9 percentage points (pp) relative to the unfiltered pool. We further investigate scaling trends for dataset design. In particular, we structure DataComp in *four* different scales, where we vary the training budget and the candidate pool size from 12.8M to 12.8B samples. Expressed in GPU hours, the cost of a single training run ranges from 4 to 40,000 GPU hours on the A100 cluster we used for development. The different scales enable researchers with different resources to participate in our benchmark. Moreover, our results show that the ranking of filtering approaches is largely consistent across scale. To make DataComp a shared environment for controlled dataset experiments, we publicly release our candidate pool url index, our tooling for assembling these pools, our filtering baselines, and our code for training and evaluating models at `www.datacomp.ai`.

In DataComp, we explore several techniques for building better models from the ground

up. Despite these advances, in practice we often want to *edit* models *after* pretraining,[1] to improve performance on downstream tasks (Zhuang et al., 2020; Wortsman et al., 2022b; Matena and Raffel, 2021; Ilharco et al., 2022), mitigate biases or unwanted behavior (Santurkar et al., 2021; Lu et al., 2022; Ribeiro and Lundberg, 2022; Murty et al., 2022), align models with human preferences (Askell et al., 2021; Ouyang et al., 2022; Kasirzadeh and Gabriel, 2022; Glaese et al., 2022), or update models with new information (Zhu et al., 2020; De Cao et al., 2021; Mitchell et al., 2021, 2022).

In the second part of this dissertation, we present **new techniques for editing neural networks after pretraining**, by operating directly in their weight space (Wortsman et al., 2022b,a; Ilharco et al., 2022, 2023). This work culminates in a framework for editing models is based on *task vectors*, which encode the information necessary to do well on a given task. Inspired by recent work on weight interpolation (Frankle et al., 2020; Matena and Raffel, 2021; Li et al., 2022; Ainsworth et al., 2022; Don-Yehiya et al., 2022), we obtain such vectors by taking the weights of a model fine-tuned on a task and subtracting the corresponding pretrained weights. We show that we can edit a variety of models with *task arithmetic*—performing arithmetic operations on task vectors. For example, *negating* a vector can be used to remove undesirable behaviors or unlearn tasks, while *adding* task vectors leads to better multi-task models, or even improves performance on a single task. Finally, when tasks form an *analogy* relationship, task vectors can be combined to improve performance on tasks where data is scarce. Overall, editing models with task arithmetic is simple, fast and effective. There is no extra cost at inference time in terms of memory or compute, since we only do element-wise operations on model weights. Moreover, vector operations are cheap, allowing users to experiment quickly. With task arithmetic, practitioners can reuse or transfer knowledge from models they create, or from the multitude of publicly available models all without requiring access to data or additional training.

Altogether, this dissertation introduces several methods for building models that are more reliable and general, both during and after pretraining. Combining dataset advances with better editing techniques, we can build the next generation of multimodal models.

---

[1] We use the term *editing* to refer to any intervention done to a model done after the pretraining stage.

Chapter 2

RELATED WORK AND BACKGROUND

2.1 Related Work

Data curation. Classical work considers dataset cleaning and outlier removal (Jiang et al., 2001; Yu et al., 2002; Rousseeuw and Hubert, 2011, 2018) to discard samples that may lead to undesirable model bias. A related line of work develops coreset selection algorithms (Har-Peled and Mazumdar, 2004; Agarwal et al., 2004; Feldman et al., 2011; Bachem et al., 2015; Lucic et al., 2018; Wei et al., 2015; Cohen et al., 2017), which aim to select data subsets that lead to the same performance as training on the entire dataset. These techniques appear to scale poorly to larger data regimes (Guo et al., 2022; Abbas et al., 2023). More recent efforts in subset selection often operate on already curated datasets (Mirzasoleiman et al., 2020; Toneva et al., 2018; Sener and Savarese, 2018; Birodkar et al., 2019; Coleman et al., 2020; Paul et al., 2021) (e.g., CIFAR-10, ImageNet) or on smaller data regimes (e.g., YFCC-15M (Radford et al., 2021; Thomee et al., 2016b)). These settings often do not reflect newer training paradigms that involve (1) *noisy* image-text pairs instead of category labeled images and (2) large scale datasets (e.g., billions of samples). While data-centric investigations have led to community competitions like DCBENCH (Eyuboglu et al., 2022) and DATAPERF (Mazumder et al., 2022), existing benchmarks have likewise operated at small data scales (Ng et al., 2021) compared to datasets like LAION-2B (Schuhmann et al., 2022), which contains over two billion images. DATACOMP bridges this gap by aligning data-centric investigation with large scale image-text training.

There has also been renewed interest in dataset pruning and deduplication. Sorscher et al. (2022) show that data pruning can improve traditional scaling trends on ImageNet, but do not consider image-text training or larger datasets. Raffel et al. (2020b) remove sentence redundancies when creating the C4 corpus. Subsequent work further demonstrated the benefits of deduplication for better language modeling (Lee et al., 2021). Radenovic et al.

(2023) introduce CAT filtering for image-text datasets—a rule-based system to retain high quality samples. Abbas et al. (2023) introduce SemDeDup, which starts with the CAT-filtered LAION-440M subset, further employing clustering to remove semantic duplicates. DataComp facilitates data-centric investigation at an even larger scale (i.e., 12.8B sample scale) and provides a common experimental setting for fair comparison amongst dataset creation algorithms.

Large-scale multimodal datasets. Datasets have been instrumental to building multimodal models like CLIP (Radford et al., 2021), Flamingo (Alayrac et al., 2022), Stable Diffusion (Rombach et al., 2022), DALL-E (Ramesh et al., 2021, 2022) and GPT-4 (OpenAI, 2023). These methods succeeded by training on large, heterogeneous datasets rather than solely through advanced modelling techniques. For example, OpenAI's CLIP trains on 400M image-text pairs from the web, roughly 300× the size of ImageNet (Deng et al., 2009). Prior work on scaling image-text datasets also provides promising trends with respect to zero-shot model performance (Jia et al., 2021; Pham et al., 2021b). Additional large scale datasets like FILIP-300M (Yao et al., 2022), FLD-900M (Yuan et al., 2021), and PaLI-10B (Chen et al., 2022) were constructed to train multimodal models. However, many datasets used to train such models (including the dataset for OpenAI's CLIP) are proprietary, making it hard to conduct data-centric investigations.

Even for public image-text datasets like SBU (Ordonez et al., 2011), Flickr30k (Young et al., 2014), MS-COCO (Chen et al., 2015), Conceptual Captions (Sharma et al., 2018), CC12M (Changpinyo et al., 2021), RedCaps (Desai et al., 2021), WIT (Srinivasan et al., 2021), Shutterstock (Nguyen et al., 2022), YFCC-100M (Thomee et al., 2016b), COYO-700M (Byeon et al., 2022), LAION-400M (Schuhmann et al., 2021), or LAION-2B (Schuhmann et al., 2022) little is known about what constitutes a good image-text dataset. Preliminary analysis suggests that different image-text data sources lead to CLIP models with different properties (Nguyen et al., 2022). However, previous work is limited to smaller scale data (10-15M examples).

The loss landscape and interpolating weights. The geometry of neural network loss surfaces has attracted the interest of several authors in recent years (Li et al., 2018; Garipov et al., 2018; Draxler et al., 2018; Kuditipudi et al., 2019; Fort et al., 2019; Czarnecki et al., 2019; Wortsman et al., 2021; Benton et al., 2021; Entezari et al., 2022; Li et al., 2022; Lubana et al., 2022). Despite neural networks being non-linear, previous work has empirically found that interpolations between the weights of two neural networks can maintain their high accuracy, provided these two neural networks share part of their optimization trajectory (Frankle et al., 2020; Izmailov et al., 2018; Neyshabur et al., 2020; Fort et al., 2020; Wortsman et al., 2022a; Choshen et al., 2022; Ilharco et al., 2022).

In the context of fine-tuning, accuracy increases steadily when gradually moving the weights of a pretrained model in the direction of its fine-tuned counterpart (Wortsman et al., 2022b; Matena and Raffel, 2021; Ilharco et al., 2022). Beyond a single task, Matena and Raffel (2021); Ilharco et al. (2022) found that when multiple models are fine-tuned on different tasks from the same initialization, averaging their weights can improve accuracy on the fine-tuning tasks. Similar results were found by Li et al. (2022) when averaging the parameters of language models fine-tuned on various domains. Choshen et al. (2022) showed that "fusing" fine-tuned models by averaging their weights creates a better starting point for fine-tuning on a new downstream task. Wortsman et al. (2022a) found that averaging the weights of models fine-tuned on multiple tasks can increase accuracy on a new downstream task, without any further training. These findings are aligned with results shown in our work. In this work, we go beyond interpolating between models, examining extrapolating between models and additional ways of combining them.

Model editing. Considering that re-training models is prohibitively expensive in most circumstances, several authors have studied more efficient methods for modifying a model's behavior with interventions after pretraining, referring to this process by different names, such as patching (Goel et al., 2020; Sung et al., 2021; Ilharco et al., 2022; Murty et al., 2022), editing (Santurkar et al., 2021; Mitchell et al., 2021, 2022), aligning (Ouyang et al., 2022; Askell et al., 2021; Kasirzadeh and Gabriel, 2022; Glaese et al., 2022), or debugging (Ribeiro and Lundberg, 2022; Geva et al., 2022). In contrast to previous literature, our

work provides a unique way of editing models, where capabilities can be added or deleted in a modular and efficient manner by re-using fine-tuned models. Closer to our work is that of Subramani et al. (2022), who explore steering language models with vectors added to its hidden states. In contrast, our work applies vectors in the weight space of pretrained models and does not modify the standard fine-tuning procedure.

2.2 Background

This section presents a brief overview of relevant concepts studied in this thesis, including distributional robustness and CLIP models (Radford et al., 2021).

Distributional robustness. One of the foundational goals of machine learning is to develop models that work reliably across a broad range of data distributions. Taori et al. (2020) categorized distribution shifts into two broad categories: (i) *synthetic*, e.g., ℓ_∞-adversarial examples or artificial changes in image contrast, brightness, etc. (Hendrycks and Dietterich, 2019; Biggio et al., 2013; Biggio and Roli, 2018; Geirhos et al., 2018; Alcorn et al., 2019); and (ii) *natural*, where samples are not perturbed after acquisition and changes in data distributions arise through naturally occurring variations in lighting, geographic location, crowdsourcing process, image styles, etc. (Taori et al., 2020; Recht et al., 2019b; Hendrycks et al., 2021a,b; Koh et al., 2021). Following Radford et al. (2019), our focus is on natural distribution shifts as they are more representative of the real world when no active adversary is present.

To measure robustness, one can contrast model accuracy on two related but different distributions, a reference distribution $\mathcal{D}_{\text{ref}}$ which is the target for training or fine-tuning, and shifted distribution $\mathcal{D}_{\text{shift}}$.[1] We assume both distributions have test sets for evaluation, and $\mathcal{D}_{\text{ref}}$ has an associated training set $\mathcal{S}_{\text{ref}}^{\text{tr}}$ which is typically used for training or fine-tuning. The goal for a model is to achieve both high accuracy and consistent performance on the two distributions $\mathcal{D}_{\text{ref}}$ and $\mathcal{D}_{\text{shift}}$. This is a natural goal as humans often achieve similar accuracy across the distribution shifts in our study (Shankar et al., 2020).

[1] $\mathcal{D}_{\text{ref}}$ and $\mathcal{D}_{\text{shift}}$ are sometimes referred to as *in-distribution* (ID) and *out-of-distribution* (OOD).

For a model f, we let $\mathsf{Acc}_{\text{ref}}(f)$ and $\mathsf{Acc}_{\text{shift}}(f)$ refer to classification accuracy on the reference and shifted test sets, respectively. We consider k-way image classification, where x_i is an image with corresponding label $y_i \in \{1, ..., k\}$. The outputs of f are k-dimensional vectors of non-normalized class scores.

Effective robustness and scatter plots. To compare the robustness of models with different accuracies on the reference distribution, we follow the *effective robustness* framework introduced by Taori et al. (2020). Effective robustness quantifies robustness as accuracy *beyond a baseline* trained only on the reference distribution. A useful tool for studying (effective) robustness are scatter plots that illustrate model performance under distribution shift (Recht et al., 2019b; Taori et al., 2020). These scatter plots display accuracy on the reference distribution on the x-axis and accuracy under distribution shift on the y-axis, i.e., a model f is shown as a point $(\mathsf{Acc}_{\text{ref}}(f),\ \mathsf{Acc}_{\text{shift}}(f))$. For the distribution shifts we study, accuracy on the reference distribution is a reliable predictor of accuracy under distribution shift (Taori et al., 2020; Miller et al., 2021). In other words, there exists a function $\beta : [0, 1] \to [0, 1]$ such that $\mathsf{Acc}_{\text{shift}}(f)$ approximately equals $\beta(\mathsf{Acc}_{\text{ref}}(f))$ for models f trained on the train set $\mathcal{S}^{\text{tr}}_{\text{ref}}$. Effective robustness (Taori et al., 2020) is accuracy beyond this baseline, defined formally as $\rho(f) = \mathsf{Acc}_{\text{shift}}(f) - \beta(\mathsf{Acc}_{\text{ref}}(f))$.

In the corresponding scatter plots, effective robustness is vertical movement above expected accuracy under distribution shift. Effective robustness thereby disentangles accuracy changes on the reference distribution from the effect of robustness interventions. When we say that a model is robust to distribution shift, we mean that effective robustness is positive. Taori et al. (2020) observed that no algorithmic robustness intervention consistently achieves substantial effective robustness across several distribution shifts—the first method to do so was zero-shot CLIP. Empirically, when applying logit (or probit) axis scaling, models trained on the reference distribution approximately lie on a linear trend (Taori et al., 2020; Miller et al., 2021).

Contrastive Language-Image Pretraining (CLIP). We primarily study CLIP models (Radford et al., 2021), which are trained to contrast images and text. Due to their text

interface, CLIP models can be used for any downstream classification task without the need to fine-tune (i.e., in a zero-shot setting). Zero-shot models exhibit effective robustness and lie on a qualitatively different linear trend in effective robustness plots. CLIP models are pretrained using image-caption pairs from the web. Given a set of image-caption pairs $\{(x_1, s_1)..., (x_B, s_B)\}$, CLIP-like models train an image-encoder g and text-encoder h such that the similarity $\langle g(x_i), h(s_i)\rangle$ is maximized relative to unaligned pairs.[2] CLIP-like models perform zero-shot k-way classification given an image x and class names $C = \{c_1, ..., c_k\}$ by matching x with potential captions. For instance, using caption $s_i =$ "a photo of a $\{c_i\}$" for each class i, the zero-shot model predicts the class via $\arg\max_j \langle g(x), h(s_j)\rangle$.[3] Equivalently, one can construct $\mathbf{W}_{\text{zero-shot}} \in \mathbb{R}^{d\times k}$ with columns $h(s_j)$ and compute outputs $f(x) = g(x)^\top \mathbf{W}_{\text{zero-shot}}$.

[2]More precisely, the image encoder g and text encoder v are trained with the loss $\ell = \frac{1}{2}\sum_{i=1}^{B} \frac{\sigma_{ii}}{\sum_{j=1}^{B}\sigma_{ij}} + \frac{1}{2}\sum_{i=1}^{B} \frac{\sigma_{ii}}{\sum_{j=1}^{B}\sigma_{ji}}$, where $\sigma_{ij} = \exp\langle g(x_i), h(y_j)\rangle$. Typically, a learnable temperature parameter is also used as in Radford et al. (2021).

[3]For improved accuracy, the embedding of a few candidate captions are averaged, e.g., $s_i^{(1)}$="a *photo* of a $\{c_i\}$" and $s_i^{(2)}$="a *picture* of a $\{c_i\}$" (referred to as prompt ensembling (Radford et al., 2021)).

Chapter 3

BUILDING BETTER MULTIMODAL DATASETS

In this chapter, we explore how data affects the resulting models. We start by showing that data is the main component responsible for the distributional robustness in CLIP models 3.1 based on findings from Fang et al. (2022). We then dive into DataComp, a benchmark for exploring dataset design in the context of CLIP pretraining 3.2, paving the way for building better multimodal datasets (Gadre et al., 2023).

3.1 Data Determines Distributional Robustness

Large pretrained multimodal models such as CLIP (Radford et al., 2021), ALIGN (Jia et al., 2021), and BASIC (Pham et al., 2021a) show unprecedented robustness on a variety of natural distribution shifts. In contrast to prior models that are trained on images with class annotations, CLIP and relatives[1] are directly trained on images and their corresponding unstructured text from the web. The resulting models achieve large robustness even on challenging distribution shifts such as ImageNetV2 (Recht et al., 2019a) and ObjectNet (Barbu et al., 2019). No prior algorithmic techniques had enhanced robustness on these datasets even after multiple years of intensive research in reliable machine learning (Djolonga et al., 2021; Taori et al., 2020). As CLIP also improves robustness on a wide range of other distribution shifts, an important question emerges: *What causes CLIP's unprecedented robustness?*

The fact that language-image models were the first to achieve large robustness gains suggests that multimodal learning on language and image data may be key to more robust image representations. However, pinpointing the exact cause of CLIP's robustness is complicated by the fact that CLIP relied on several changes to the common supervised training

[1]Following Radford et al. (2021), we use *CLIP* as a name for the general training technique, not only their specific models.

paradigm for image classification models. For instance, the CLIP models with highest accuracy follow the vision transformer (ViT) architecture (Dosovitskiy et al., 2021). Radford et al. (2021) already investigated model architecture and size, showing that these factors do not affect the robustness of their CLIP models. Nevertheless, there is still a long list of possible causes for CLIP's robustness:

- The large training set size (400 million images)
- The training distribution
- Language supervision at training time
- Language supervision at test time via prompts
- The contrastive loss function

Understanding the mechanism underlying CLIP's robustness is important as it may guide the way towards more reliable machine learning more broadly.

In Fang et al. (2022), we answer the question of CLIP's robustness via a series of controlled experiments that test the five possible causes listed above. Our main result is that CLIP's robustness is determined almost exclusively by the training distribution. Language supervision at training time does *not* make the resulting models more robust than standard supervised learning when the images in the training set are the same. Hence language supervision only has an *indirect* effect on robustness. In particular, language supervision simplifies training on a diverse distribution of images by removing the need for consistent annotation with class labels. The more diverse training distribution-—not the language supervision-—then leads to more robust representations.

Our investigation of CLIP's robustness rests on two further contributions. First, we introduce ImageNet-Captions, a new dataset for training on paired language-image data. ImageNet-Captions augments 463,622 of the 1.2 million images in the ImageNet 2012 training set (Deng et al., 2009) with the original text data sourced from the corresponding Flickr images. ImageNet-Captions enables controlled experiments comparing standard (class-based) ImageNet training with language-image training on the same set of images. Such experiments precisely pinpoint the effect of utilizing language when training computer vision models.

Second, we provide a new baseline for language-image training that minimizes the interaction between the vision and language components yet achieves accuracy similar to CLIP training. Specifically, we introduce the following training procedure and illustrate its behavior on the YFCC-15M dataset (Thomee et al., 2016a; Radford et al., 2021): 1) Use SimCLR (Chen et al., 2020a) to pretrain a representation on only the *images* in YFCC-15M; 2) Finetune the resulting representation by matching examples in YFCC-15M to ImageNet classes with simple *text matches* in the corresponding captions.

Our approach relies on *no* language model, demonstrating that it is possible to match the performance of CLIP training with much simpler language processing. Besides serving as a useful baseline to understand CLIP training, our simplified approach may open the way for further algorithmic innovations in language-image training.

3.1.1 ImageNet-Captions

We now describe ImageNet-Captions, our new dataset for experiments with image-text supervision.[2] Four desiderata guided the creation of ImageNet-Captions:

1. To isolate the effect of natural language supervision on effective robustness, we require a dataset that contains *both* natural language supervision and traditional classification labels. This setup allows us to train classifiers separately with contrastive image-text losses and with standard classification losses on the *same images* and compare the resulting models. Differences in the models are then solely due to different loss functions, not architectural differences or different training distributions.
2. The text annotations in the dataset should come from the original image source, as opposed to synthetically generated captions from curated templates or an image captioning model. This helps ensure that the dataset is representative of image-text data "in the wild" and minimizes artifacts from templates or machine models.
3. The dataset should be related to commonly studied benchmarks, such as ImageNet, in order to have good baselines and comparable training methods.
4. The dataset should be large enough to support training on modern neural networks.

[2]The dataset is available at https://github.com/mlfoundations/imagenet-captions.

We constructed ImageNet-Captions to satisfy all four desiderata. ImageNet-Captions is a subset of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 training set, paired with the original image title, description, and tags from Flickr (recall that a large part of ImageNet was sourced from the Flickr image hosting website).

Since ImageNet is a widely used image classification benchmark, our goal was to augment the 2012 ImageNet training set with original text data. A priori, this is a difficult task since the standard 2012 ImageNet release does not contain any metadata for the images. As a starting point, we leveraged three facts about ImageNet: 1) A large fraction of ImageNet is sourced from Flickr; 2) The ImageNet fall 2011 release contained URLs for each image in the full ImageNet dataset; 3) For a given photo identifier, the Flickr API provides the associated text data.

Our dataset construction began with filtering the 14,197,122 image URLs in the ImageNet fall 2011 release to only include images from Flickr. In addition, we restricted the images to just the 1,000 classes included in the 2012 ImageNet competition (every entry in the fall 2011 release contains both a URL and a class label). After this filtering, we were left with 642,147 images belonging to 999 classes (all classes in ILSVRC-2012 except "teddy bear").

Next, we ran the image deduplication routine of Jain et al. (2019) to remove images that were not in the ILSVRC-2012 training set. In addition, we removed text containing profanity. This left us with a dataset of 463,622 images that are in the ILSVRC-2012 training set, along with the newly obtained corresponding text data. In particular, for each image we extracted a title (the text at the top of the Flickr image), description (the text at the bottom of the Flickr image), and user-provided tags. Since these images are a subset of ILSVRC-2012, we also have a corresponding class label that can be used for standard ImageNet training.

The resulting dataset contains captions from a mix of 127 different languages with the bulk (90%) coming from English. We further inspected the quality of image-text pairs by checking for the presence of the desired class label in the associated text. Table 3.1 summarizes the analysis. We find that for 94% of the images, the name of the ImageNet class is present in the corresponding text. This indicates that most of the captions contain

Table 3.1: Images from ImageNet-Captions contain three types of metadata: titles, description, and tags. For each type of metadata, this table shows the number of images that have corresponding metadata that contains the class label of the image. For most images, the class label is in at least one text field, indicating that ImageNet-Captions is suitable for language-image training.

Caption Type	# Images	% of Total
Title Only	239,495	51.6
Description Only	134,387	28.9
Tags Only	342,340	73.8
Title, Tag and Description	435,239	93.8

relevant information about the class and are suitable for training image-text models.

3.1.2 ImageNet-Captions experiments

In this section, we use the ImageNet-Captions dataset to investigate the effect of language on robustness. ImageNet-Captions provides a simple comparison with vision-only methods because ImageNet is considered the premier benchmark for image classification. We train the ResNet-50 based CLIP model on ImageNet-Captions with a contrastive loss, as well as the vision encoder of that CLIP model with an additional linear layer on the equivalent image classification dataset.

Caption constrution. When constructing ImageNet-Captions, we had to choose which parts of the metadata to include in the caption. To do so, we ran experiments on variants that included just the title, the title followed by the description, and the title followed by the tags followed by the description. Furthermore, Radford et al. (2021) use a filter to keep only images with captions in English. We create additional variants of the dataset by applying a similar filter. As shown in Table 3.2, captions that include more information appear to perform better. Furthermore, it seems that filtering for cleaner captions does not

Table 3.2: Evaluating different caption variants across ImageNet (IN) natural distribution shifts. Results are reported in top-1 accuracy (%). The best performing caption uses title, tags, and description. Although the language filter makes captions cleaner, the decrease in overall dataset size decreases performance.

Title	Desc	Tags	Filter	Size	Relative Size (%)	IN	IN-V2	IN-R	IN Sketch	ObjectNet	IN-A
✓			✓	197K	42.6	15.7	12.2	6.6	1.1	5.5	2.4
✓				459K	99.0	26.2	20.7	9.5	2.6	8.4	2.7
✓	✓		✓	312K	67.4	21.9	16.5	8.0	1.7	6.1	2.2
✓	✓			461K	99.4	27.8	21.6	9.6	3.0	8.0	2.7
✓	✓	✓	✓	367K	79.3	26.5	20.3	8.9	2.3	7.9	2.5
✓	✓	✓		464K	100.0	31.5	24.0	10.9	2.7	9.1	3.0

make up for the loss of image-caption pairs. In caption construction ablations, images with empty captions were dropped, causing variation in dataset size across the experiments in Table 3.2.

Robustness. To determine the robustness of models trained on ImageNet-Captions, we evaluate on ImageNet and compare with natural distribution shifts in ImageNetV2, ImageNet-R, ImageNet Sketch, ObjectNet, and ImageNet-A. In Figure 3.3, we see that ImageNet-Captions CLIP models roughly follow the same linear trends as ImageNet-Captions classification models across the various distribution shifts. This shows that CLIP models are not more robust than classification models trained on the same dataset, despite the difference of language supervision. This is a better comparison than that with ImageNet classification models because there is no longer the potential confounding factor of the datasets having different image distributions. Nevertheless, these models do not achieve the robustness seen in CLIP models from Radford et al. (2021).

Pretraining on language. While the above experiments show that language supervision from ImageNet-Captions does not contribute to a model's robustness, it does not rule out

robustness coming from the language supervision of OpenAI's proprietary dataset used to train CLIP. Therefore we ran additional experiments where we loaded the pretrained OpenAI CLIP model onto the language encoder, while randomly initializing the vision encoder. We trained ImageNet-Captions on this setup, with an additional variant where we also freeze the language encoder's weights. As seen in Figure 3.4, while both the unfrozen and frozen variants of the pretrained language encoder increased the accuracy of the model when compared to the completely randomly initialized model, neither variant provided additional effective robustness.

Effect of using templates. Given that images in ImageNet-Captions have a corresponding ImageNet class, we can try to leverage this information to investigate the effect of captions and class information on both accuracy and robustness. Radford et al. (2021) introduces prompt templates in formats similar to "A photo of a {label}." Creating templates for ImageNet-Captions is different than doing so for other image-text datasets because each image already has an assigned label; for other datasets, creating a template requires looking through the caption for classes, which are not guaranteed to be in the caption.

We found that attaching templates at the beginning of captions (followed by Title + Tags + Description) achieves 34.7% ImageNet top-1 accuracy, which is 3.2% more than without the templates. However, using the templates by themselves as the captions achieves 50.5% ImageNet top-1 accuracy, suggesting that the additional information in the captions hurts ImageNet performance. The model trained on the equivalent classification task achieves 48.7%, which suggests that with additional parameter tuning, classification may be similar to CLIP training on templates.

While using templates instead of captions can increase ImageNet performance, it does not improve robustness. Using templates on top of the captions follows linear trends similar to ImageNet-Captions. In fact, training a model on all of ImageNet using templates behaves like an equivalent classification model.

Improving ImageNet performance using captions. While language supervision does not improve robustness, it is still possible that the additional information may improve

ImageNet accuracy. We investigate this by running experiments on ImageNet, augmented with ImageNet-Captions. It is well known that ResNet-50 achieves 77.15% top-1 ImageNet accuracy (He et al., 2016). As a similar baseline, we achieve 76.62% top-1 ImageNet accuracy by training the CLIP model with templates as the caption.

We have tried improving this baseline by initializing the language head with the OpenAI pretrained model, using the combined ImageNet (templates) and ImageNet-Captions for training, using ImageNet-Captions as text augmentation when available, and contrasting the image encoding with both the template and the ImageNet-Captions caption encodings when available. However, all of these fall within ±1% of the baseline. Note that concatenating the captions to the templates changes the image distributions, while some of the other approaches do not. On the other hand, restricting the images used to those within ImageNet-Captions hints that language may help improve ImageNet performance. The experiments we have run are non-exhaustive, and we leave it to future work to find whether language information can improve ImageNet performance, and more broadly, vision task performance.

3.1.3 YFCC experiments

Our experiments in the previous section show that language supervision alone does not improve robustness. To further understand the source of CLIP's robustness, we now investigate whether it is possible to train a representation with minimal or even no language supervision that still yields the same robustness as CLIP. These results will provide further evidence that CLIP's robustness stems from the more diverse data distribution, not the presence of language supervision.

Our experiments in this section start with a language-image training set on which CLIP exhibits improved robustness: the Yahoo Flickr Creative Commons dataset (YFCC) (Thomee et al., 2016a). To test whether the image data in YFCC alone can improve robustness, we contrastively pre-train a "standard" image representation on YFCC that does not involve the language part of the dataset. Building on this image-only representation, we then train a zero-shot classifier with only minimal text processing (substring matches). The resulting classifier achieves effective robustness close to CLIP. This demonstrates that the training

Table 3.3: Comparing CLIP training with (language model free) classification models on YFCC-15M. All experiments use a ViT-B/16 backbone. The CLIP results are from Mu et al. (2021). Image-only contrastive learning followed by a simple text matching stage for classification nearly matches the performance of CLIP with a full language model.

Training style	ImageNet	Avg OOD
CLIP	37.9	19.9
SimCLR → Classification	35.7	18.8

distribution, not language supervision at training time, is the main reason behind CLIP's robustness.

Dataset. We use the YFCC-15M (Radford et al., 2021) dataset, a subset of YFCC-100M (Thomee et al., 2016a) filtered to only images with English titles or descriptions. The dataset contains 14,829,396 images with natural language captions associated with each image. To train image classifiers on YFCC-15M, we convert YFCC-15M into a classification dataset with class labels for each image, which we denote YFCC-15M-Cls. We assign ImageNet labels to each image using a simple strategy: if the title or description contains the name of an ImageNet synset or synonym (Miller, 1995), we assign the corresponding synset label to the image. If an image contains no or multiple ImageNet synsets, we discard that image. This results in 1,694,125 images (11.4% of the full dataset) covering 953 ILSVRC classes. The least common class has 1 image, while the most common has 280,351 images.

Classification training. We use a ViT-Base (ViT-B/16) model fine-tuned using the softmax cross-entropy loss on YFCC-15M-Cls. Since this data is only a fraction of YFCC-15M, we initialize the classification model with a SimCLR model pre-trained on YFCC-15M from Mu et al. (2021).

Results. We present our main results in 3.1. A CLIP model trained on all images and captions from YFCC-15M yields an ImageNet top-1 accuracy of 37.9%. Our baseline classification model[3], which trains SimCLR on YFCC-15M, but fine-tunes on only a small fraction (about 11%) of the supervision in YFCC-15M, results in an accuracy of 35.7%, which we found surprisingly close to CLIP. Further, as shown in 3.1 ("YFCC SimCLR + Classification"), our baseline model's effective robustness is similar to that of CLIP.

Overall, we find that despite largely eschewing language, and training on a fraction of the supervision, our baseline model results in high effective robustness, similar to CLIP. These results indicate that image-only pre-training followed by classification fine-tuning can match the robustness of CLIP, and that language pre-training is *not* necessary for effective robustness. Models trained on YFCC consistently achieve higher effective robustness than models trained on ImageNet, which shows that different training distributions have different levels of effective robustness.

3.1.4 Effect of test time prompts

As another hypothesis, we study whether natural language prompts affect CLIP's robustness. Recall that prompts consist of a template (e.g., *"a photo of ___"*) and the name of a class in the dataset. Radford et al. (2021) showed how to use multiple templates by averaging their text representations. Similarly, it is also possible to use multiple class names for each class if synonyms exist (e.g. *microwave* and *microwave oven*). To investigate the influence of specific prompts in the robustness of CLIP, we conduct a series of experiments using a trained CLIP model and multiple prompting strategies. Specifically, we vary:

- The templates used, using one of the following three options:

  i) Templates from Radford et al. (2021);

  ii) No templates (i.e., only the class names);

  iii) Random words appended before and after the class name.[4]

[3]We call this baseline "NoCLIP", for "**No**w we use SimCLR+Classification instead of **C**ontrastive **L**anguage-**I**mage **P**re-training

[4]Templates are composed by one to ten random words along with the class name, in an arbitrary position. Random words are drawn using `https://pypi.org/project/Random-Word/`.

- The names of the classes, using one of the following three sources:

  i) Class names from Radford et al. (2021);

  ii) Class names from WordNet synset (Miller, 1995);

  iii) A combination of the previous two sources.
- The number of templates used, chosen from $\{1, 2, 4, 8, 16, 32, 80\}$.
- The maximum synonyms per class, one of $\{1, 2, 4\}$.

Figure 3.5 (left) shows the results from over a hundred experiments. We find that specific choices of prompts can have a substantial impact on performance. While some prompt variations did increase effective robustness, this increase is entirely due to the substantially reduced accuracy. In particular, one can achieve the same change in effective robustness by simply interpolating with a random classifier (which sees no performance change under distribution shift). We illustrate this behavior with the brown line in Figure 3.5 (left). Overall, these results show that prompts are not the source of robustness of CLIP models.

3.1.5 Effect of contrastive training losses

Finally, we explore contrastive pre-training as a potential source of CLIP's robustness. Contrastive pre-training is a popular method for self-supervised representation learning that encourages similar pairs to be close and dissimilar pairs to be far apart in a learned representation space. In Radford et al. (2021), the similar pairs are images and their corresponding captions. In SimCLR (Chen et al., 2020a), similar pairs are the same images with different data augmentation.

As the contrastive loss is core to CLIP's approach, we explore whether contrastive approaches independently promote effective robustness. Figure 3.5 (right) shows results for various popular contrastive methods, including SimCLRv2 (Chen et al., 2020b), SimSiam (Chen and He, 2020) and SwAV (Caron et al., 2020), pre-trained on ImageNet. We evaluate on ImageNet and the five distribution shifts. While the methods differ significantly from each other (e.g., different augmentation strategies, memory banks, and feature clustering techniques), they consistently exhibit little to no effective robustness.

3.1.6 Conclusion

The previous sections have systematically ruled out the training set size, language supervision, and the contrastive loss function as explanations for the large robustness gains achieved by the CLIP models of Radford et al. (2021). In addition, Section 3.1.3 has demonstrated that changing the training distribution from ImageNet(-Captions) to YFCC substantially affects the robustness of the resulting models. We arrive at a clear conclusion: CLIP's robustness is dominated by the choice of the training distribution, with other factors playing a small or non-existent role. While language supervision is still helpful for easily assembling training sets, it is not the primary driver for robustness. Overall, our results add to a growing body of evidence that the training distribution plays a central role for mitigating real-world distribution shifts.

3.2 DataComp: In Search of the Next Generation of Multimodal Datasets

As shown in the previous section, image-text datasets play a crucial role in building multimodal models. In this section, we introduce DataComp (Gadre et al., 2023), a new benchmark for dataset design, created to help advance our understanding on how to create good pretraining multimodal datasets.

DataComp is meant to facilitate data-centric experimentation. While traditional benchmarks emphasize model design, DataComp is centered around dataset development, where the resulting datasets can be used to train high accuracy models. We focus on large image-text datasets and quantify a dataset submission by training a CLIP model on it from scratch (Radford et al., 2021) and evaluating on 38 downstream image classification and retrieval tasks. We additionally have three secret test sets, which will be released after a year, to guard against overfitting. To facilitate such investigations, we provide a candidate pool of uncurated image-text pairs sourced from the public internet. Our benchmark offers two tracks: one where participants must filter samples from the pools we provide, and another where participants can use external data. Moreover, DataComp is structured to accommodate participants with diverse levels of computational resources: each track is broken down into four scales with varying compute requirements. We now discuss high-level de-

sign decisions, construction of a 12.8B image-text data pool to facilitate the competition, benchmark tracks, model training, and evaluation.

3.2.1 Competition design

In many areas of machine learning, larger datasets can lead to better performing models (Krizhevsky et al., 2012; Kaplan et al., 2020; Jia et al., 2021; Pham et al., 2021b; Hoffmann et al., 2022a; Cherti et al., 2022; Brown et al., 2020a; Radford et al., 2021, 2022). Hence comparing only datasets with the same size is a natural starting point. However, this approach is flawed as controlling the dataset size ignores critical curation constraints: candidate pool size (i.e., number of image-text pairs to harvest) and training compute. For instance, assembling a dataset like LAION-2B consists of identifying *data sources* (e.g., Common Crawl or Reddit) and *filtering* the data source. Notably, *the final dataset size is a design choice* and is only upper-bounded by the data sources. Hence, the true data constraint is the size of the reservoir of samples: *candidate pool* to be filtered. To make DataComp a realistic benchmark, we therefore fix the candidate pool in the filtering track, but give participants control over the training set size.

Compute cost is another relevant constraint. To put datasets of different size on equal footing, we specify the total *number of training samples seen*. Consider the 12.8B compute scale and filtered datasets A and B, with 6.4B and 3.2B image-text pairs respectively. At this scale, we train by making two passes over A, while making four passes over B. A key result from our experiments is that smaller, more stringently filtered datasets can lead to models that generalize *better*.

Competition tracks. Two key procedures in assembling a training dataset are filtering a data source (Schuhmann et al., 2021, 2022; Byeon et al., 2022) and aggregating data sources (Dave et al., 2020; Deng et al., 2009). To reflect this structure, DataComp has two tracks: *filtering*, where participants select a subset of the samples from CommonPool, and *Bring Your Own Data* (BYOD), where participants can use any source of data. Key decisions for each tracks are described in Sections 3.2.2 and 3.2.3, respectively.

Table 3.4: Experimental configurations, with compute in multiply-accumulate operations (MACs).

Scale	Model	Train compute (MACs)	Pool size and # samples seen
`small`	ViT-B/32	9.5×10^{16}	12.8M
`medium`	ViT-B/32	9.5×10^{17}	128M
`large`	ViT-B/16	2.6×10^{19}	1.28B
`xlarge`	ViT-L/14	1.1×10^{21}	12.8B

Competition compute scales. To facilitate study of scaling trends and accommodate participants with various computational resources, we structure DataComp using four scales of compute: `small`, `medium`, `large` and `xlarge`. Each new scale increases the number of samples seen during training by 10× (from 12.8M to 12.8B samples seen), and the pool we provide by the same factor (from 12.8M samples to 12.8B samples). Table 3.4 gives the experimental configuration used for each scale. For the `small` scale, our runs took 4 hours on an A100 GPU, and for the `xlarge` scale 81 hours on 512 GPUs.

3.2.2 CommonPool *generation, for the filtering track*

We construct a large-scale pool of image-text pairs, CommonPool, from Common Crawl. CommonPool is distributed as an image url-text pair index under a CC-BY-4.0 license. Our pool construction pipeline has four steps: url extraction and data download, NSFW detection, evaluation set deduplication, and face blurring. We additionally provide per sample metadata (e.g., CLIP features). Starting from the `xlarge` CommonPool, we take successive random subsets to create `large`, `medium`, and `small` CommonPool (e.g., `medium` is a subset of `large`).

Extracting urls and dowloading data. We first use `cc2dataset`[5], which utilizes Apache Spark (Zaharia et al., 2016), to extract pairs of image urls and nonempty alt-text from all

[5] https://github.com/rom1504/cc2dataset

Common Crawl snapshots from 2014 to 2022. We then deduplicate the url-text pairs and randomly shuffle. This step results in ∼88B possible samples. Not all samples are downloadable; other samples are not suitable due to NSFW content or overlap with our evaluation sets. We attempt to download ∼40B samples using `img2dataset`[6] resulting in ∼16.8B image-text pairs.

Safety preprocessing. Since Common Crawl is a snapshot of the internet, we require strict preprocessing to remove unsafe content. We use Detoxify (Hanu and Unitary team, 2020) to prune samples that contain unsafe text (e.g., obscene, sexually explicit, or threatening language). We also discard samples with explicit visual content. To do so, we train a classifier on CLIP ViT-L/14 (Radford et al., 2021) features, using the NSFW dataset used in LAION-5B (Schuhmann et al., 2022). We validate our classifier against the Google commercial image safety API. Around 19% of image-text pairs are considered NSFW, taking the pool of ∼16.8B downloads to ∼13.6B samples.

Evaluation set deduplication. To prevent accidental overfitting to certain test sets in our evaluation suite, we perform a thorough near-duplicate removal between the candidate pool and our evaluation sets, using a state-of-the-art image deduplication model (Yokoo, 2021). The model flags ∼3% of the 16.8B images as near-duplicates, reducing the ∼13.6B pool to ∼13.1B samples. From here we select a random subset to get the `xlarge` pool of 12.8B samples.

Face detection & blurring. To protect the privacy of individuals, we detect and blur faces from images in our pool using a face detector (Guo et al., 2021). As observed by Yang et al. (2022), obfuscating faces has little impact on model performance, as we also observe in our experiments.

Pool metadata. To bootstrap participants we distribute metadata for each sample in CommonPool (e.g., image url, alt-text, original image resolution, CLIP features, and

[6] https://github.com/rom1504/img2dataset

CLIP similarity scores). Following Carlini et al. (2023), we release SHA256 hashes for each image to guard against data poisoning in subsequent CommonPool downloads.

*3.2.3 The bring your own data (*BYOD*) track*

While CommonPool can be used to study different filtering techniques, state-of-the-art models often train on data from different sources. For instance, the Flamingo model (Alayrac et al., 2022) uses both multimodal massive web (M3W) and ALIGN datasets (Jia et al., 2021). To facilitate non-proprietary research on curating data from many sources, we instantiate a separate DataComp track to allow participants to combine multiple data streams. For example, participants could construct a training set from CC12M (Changpinyo et al., 2021), YFCC100M (Thomee et al., 2016b), and data sources they label themselves.

3.2.4 Training

We create a common experimental setting that enables comparable experiments by fixing the training procedure. We closely follow the CLIP training recipe proposed by Radford et al. (2021): training models from scratch with a contrastive objective over images and captions. Given a set of image-caption pairs, we train an image encoder and a text encoder such that the similarity between the representations of images and their corresponding text is maximized relative to unaligned pairs. We also use a learnable temperature parameter as in Radford et al. (2021). For each scale, we fix the model architecture and hyperparameters (see Table 3.4). We pick Vision Transformers (ViTs) (Dosovitskiy et al., 2021) as the image encoder, considering the better scaling trends observed by Radford et al. (2021) compared to ResNets (He et al., 2016). Models are trained for a fixed number of steps determined by the scale (Table 3.4), using the OpenCLIP repository (Ilharco et al., 2021).

3.2.5 Evaluation

We evaluate on a suite of 38 image classification and retrieval tasks. As discussed in Section 3.2.2, we remove test set images from DataComp to avoid contamination. Image classification datasets range from satellite imagery recognition to classifying metastatic tis-

sues. In total we have (with some overlap): 22 of the datasets evaluated in Radford et al. (2021), 6 ImageNet distribution shifts (i.e., ImageNet-Sketch (Wang et al., 2019), ImageNet-V2 (Recht et al., 2019a), ImageNet-A (Hendrycks et al., 2021c), ImageNet-O (Hendrycks et al., 2021c), ImageNet-R (Hendrycks et al., 2021a), and ObjectNet (Barbu et al., 2019)), 13 datasets from VTAB (Zhai et al., 2019), and 3 datasets from WILDS (Koh et al., 2021; Sagawa et al., 2022). Retrieval datasets include Flickr30k (Young et al., 2014), MSCOCO (Chen et al., 2015), and the WinoGAViL commonsense association task (Bitton et al., 2022). To aggregate results over all evaluation tasks, we average the preferred metric for each task.

DataComp adopts a zero-shot evaluation protocol: models are tested without training on the evaluation tasks. This approach is computationally efficient and measures a model's ability to perform well without any additional training. We find a strong rank correlation (>0.99) between performance in linear probe zero-shot settings.

3.2.6 Baselines

We study six simple filtering methods for the filtering track:

- **No filtering.** We simply use the entire pool as the subset, without any filtering. Since each pool size is equal to the sample budget, training consists of one pass over the data.
- **Random subsets.** To isolate the effects of increasing the compute budget from increasing the dataset size, we form subsets consisting of 1%, 10%, 25%, 50% and 75% of the pool chosen at random.
- **Basic filtering.** We consider many simple filtering operations inspired by Schuhmann et al. (2021) and Byeon et al. (2022): filtering by *language* (English captions, using either fasttext (Joulin et al., 2017) or cld3[7]); filtering by *caption length* (over two words and five characters); and filtering by *image size* (smaller dimension above 200 pixels and aspect ratio below three). We also experiment with combining language and caption length filtering and combining language, caption length, image size fitering. Unless otherwise specified, "basic" refers fasttext English, caption length, and image

[7] https://github.com/google/cld3

size filtering.

- **CLIP score and LAION filtering.** We experiment with CLIP score filtering (also employed by LAION), where we take only examples having cosine similarity scores between CLIP image and text embeddings that exceed a pre-defined threshold. We investigate a range of thresholds and two OpenAI CLIP models for computing the scores: the ViT-B/32 model (as in LAION) and the larger ViT-L/14. We also combine CLIP score thresholds and cld3 English filtering to reproduce the LAION-2B filtering scheme.
- **Text-based filtering.** We select examples that contain text overlapping with ImageNet class names, which serve as a proxy for relevance to downstream tasks. Specifically, we select English captions (according to fasttext) that contain words from ImageNet-21K or ImageNet-1K (Deng et al., 2009) class synsets.
- **Image-based filtering.** We select a subset of examples whose visual content overlaps with ImageNet classes. After applying English language (fasttext) and caption length filtering, we cluster the image embeddings extracted by the OpenAI ViT-L/14 model for each image into 100K groups using Faiss (Johnson et al., 2019). We then find the nearest neighbor group for every ImageNet training example, and keep examples belonging to these groups. We apply this procedure using either ImageNet-21K (14M images) or ImageNet-1K (1.2M images), forming two subsets.

We additionally experiment with multiple external data sources, including four moderately sized datasets (10 to 58M samples) studied by Nguyen et al. (2022)—CC12M (Changpinyo et al., 2021), YFCC15M (Thomee et al., 2016b; Radford et al., 2021), RedCaps (Desai et al., 2021) and Shutterstock (Nguyen et al., 2022)—and the larger LAION-2B (Schuhmann et al., 2022). We consider these data sources as they are and do not perform additional preprocessing. We also present experiments combining some of the data sources (using only the external datasets, or in addition to data from our pool).

3.2.7 Results and discussion

Main results. Our key results are in Table 4.1. Most notably, the intersection between image-based filtering and CLIP score filtering excels on most tasks. The exception is at the `small` scale and for retrieval datasets.[8] Furthermore, other filtering strategies like basic, CLIP score, image-based, text-based filtering show better downstream performance when compared to no filtering.

DataComp leads to better image-text datasets. We hope DataComp catalyzes the search for the next generation of multimodal datasets. We contribute DataComp-1B, which is the output of the Image-based ∩ CLIP score (L/14 30%) baseline filter at the `xlarge` scale of the filtering track. Our dataset is comprised of 1.4B samples, which not only is *smaller* than the LAION-2B dataset with 2.3B samples, but also comes from a smaller pool. Nevertheless, a CLIP L/14 trained on DataComp-1B outperforms the LAION-2B competitor by 6.1 percentage points on ImageNet. Moreover, training on DataComp-1B improves ImageNet accuracy by 3.7 percentage points over OpenAI's ViT-L/14 trained with the same compute budget. Additionally, even if we restrict ourselves to 400M samples, we can still find a subset of DataComp-1B that outperforms OpenAI's ViT-L/14. These results demonstrate the impact that DataComp can make and provide a foundation upon which participants can build.

External data sources can improve performance. We find several instances where adding external data sources improves performance over using just data from CommonPool. For example, at the `large` scale, combining CLIP-filtered data from CommonPool with external data from CC12M (Changpinyo et al., 2021), YFCC15M (Thomee et al., 2016b; Radford et al., 2021), RedCaps (Desai et al., 2021) and Shutterstock (Nguyen et al., 2022) boosts ImageNet accuracy by 4.3 percentage points.

Trade-off between data diversity and repetition. In Figure 3.6, we see that randomly selecting subsets of the pool has little effect and degrades performance substantially

[8] Cherti et al. (2022) also observe that models rank differently on classification and retrieval tasks.

when only small fractions are used. When filtering with CLIP scores, the optimal training set comes from selecting ∼30% of the pool with the highest scores. The difference in performance trends between random subsets and CLIP score filtering highlights the importance of filtering strategies for selecting samples.

CommonPool and LAION are comparable with the same filtering. To validate our pool construction, we show that we can build datasets comparable to LAION-2B by employing their filtering technique on our pool. LAION-2B selects all samples where the caption is in English and the cosine similarity score from a trained ViT-B/32 CLIP model is above 0.28. We compare this filtering approach on our pool using the same number samples, 130M samples at the `large` scale. We find that the different data sources perform comparably: 55.3% vs 55.7% accuracy on ImageNet, and 0.501 vs 0.489 average performance over our evaluation sets using our pool and LAION-2B, respectively.

Consistency across scales. We find that the ranking between filtering strategies is typically consistent across different scales. This is illustrated in Figure 3.7, which shows that the baselines at `small` and `medium` scales are positively correlated. Moreover, the rank correlations of performance is high, between 0.71 and 0.90 for different scale pairs.

Consistency across training changes. DataComp fixes the training procedure, so a natural question is whether better datasets from DataComp are better outside of DataComp. While DataComp-1B is trained at the `xlarge` scale, we find that even when substituting the ViT-L/14 for a ViT-B/16 or ViT-B/32, training on DataComp-1B outperforms training on OpenAI's WIT and LAION-2B. Additionally, we found that modifying hyperparameters such as training steps and batch size minimally affects the relative ordering of different data curation methods on downstream performance.

ImageNet accuracy is indicative, but not the complete picture. Similarly to Kornblith et al. (2019b), we find that ImageNet performance is highly correlated with the average

performance across all datasets we study, with an overall correlation of 0.99. [9] However, ImageNet performance is not representative of all evaluation tasks, as the correlation between ImageNet accuracy and accuracy on other individual datasets varies substantially, in some cases even exhibiting a negative correlation.

Robustness and fairness. While typical models trained on a target task suffer large performance drops under data distribution shift, zero-shot CLIP models are known to exhibit strong performance across many distributions (Radford et al., 2021). We find that CLIP models trained with data from our pool are more robust to distribution shift than ImageNet-trained models from Taori et al. (2020)'s testbed. Examining geographic diversity, we find that our models are better than ImageNet-trained models, but fall short of models fine-tuned on diverse curated datasets. We also perform a face classification analysis and identify demographic biases in our models: notably, the BYOD datasets we consider can increase the risk of misclassification.

3.2.8 Limitations and conclusion

In terms of societal risks, creating an index of image-text pairs from the public internet can be problematic. The internet contains unsafe, toxic, and sensitive content, which ideally should not percolate into machine learning datasets. Though we take steps to remove NSFW content and blur human faces to protect privacy, we hope future work will further explore the biases and risks from CommonPool and DataComp-1B. We see several additional directions for future work, including 1) Curating more data sources. 2) Improved data filtering algorithms. 3) Further supervision signals (e.g., image captions coming from captioning models). 4) Additional input modalities (e.g., video, 3D objects). 5) Broader evaluations for vision-and-language and robotics tasks.

Overall, we see DataComp as a first step towards improving training datasets, and hope our new benchmark will foster further research. By providing a controlled experimental setting, DataComp enables researchers to iterate on dataset design on rigorous empirical

[9] Note that unlike Kornblith et al. (2019b) we evaluate zero-shot performance rather than transfer learning.

foundations. We open-source all of our code, data, and infrastructure, and hope these resources will help the community build the next generation of multimodal datasets.

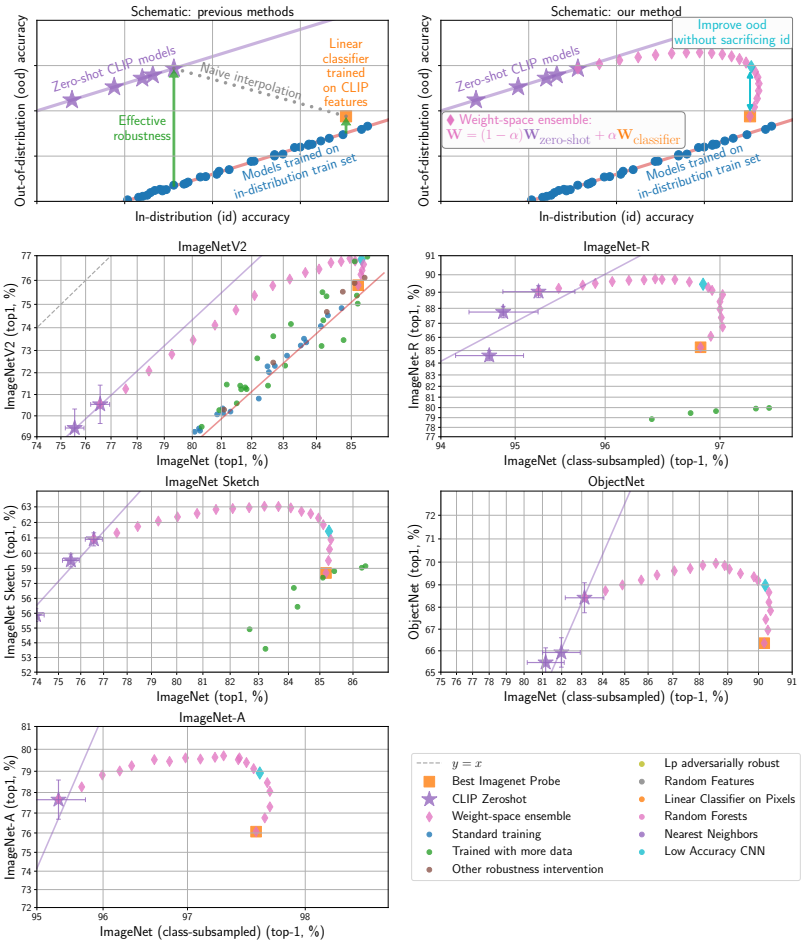

Figure 3.1: We compare models trained using different methods and on different datasets, measuring their robustness on a range of natural distribution shifts (ImageNetV2, ImageNet-R, ImageNet-Sketch, and ObjectNet). The CLIP models stand out with their consistent performance in the presence of distribution shift. We find that large gains in effective robustness (improvement over ImageNet models) only come from varying the training distribution. Language supervision alone does not cause robustness.

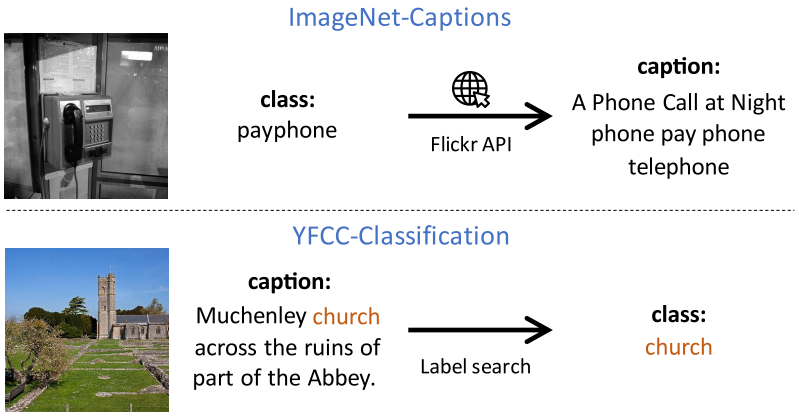

Figure 3.2: Overview of the two main training sets, Imaget-Captions and YFCC-Classification. (Top) We introduce the ImageNet-Captions dataset, where we *augment* a subset of the ImageNet 2012 training set images with the corresponding original captions collected from Flickr. (Bottom) We convert the YFCC image-caption dataset into YFCC-Classification by searching for class labels in the YFCC captions and then *removing* the text annotations. These two datasets allows us to evaluate the impact of language-image training on robustness because we can compare language-image training with standard classification training *on the same set of images*.

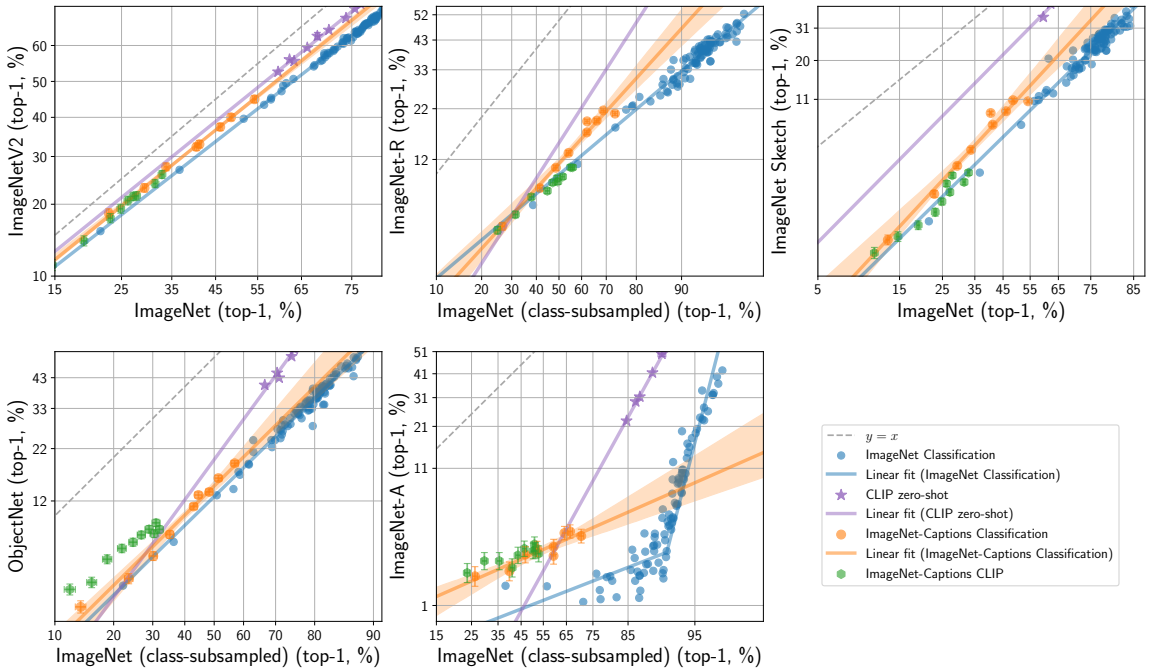

Figure 3.3: On most natural distribution shifts, models trained with language information from ImageNet-Captions follow the same trend as models trained without it. Neither comes close to achieving the robustness of OpenAI's CLIP models.

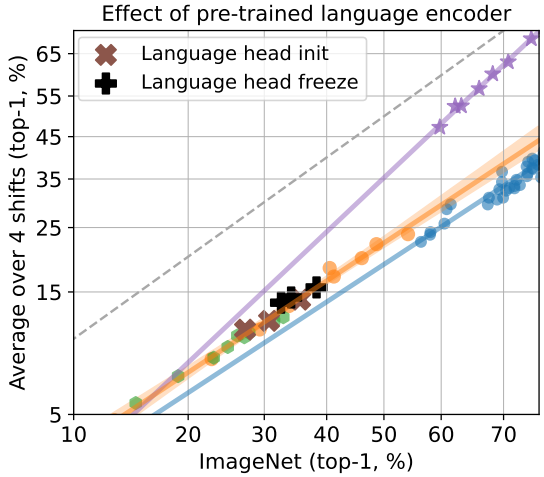

Figure 3.4: Using the weights from OpenAI's pretrained CLIP model does not improve robustness, despite the large size of the full CLIP training set (400 million images). This is further evidence that language supervision does not increase robustness. See Figure 3.3 for remaining legend elements.

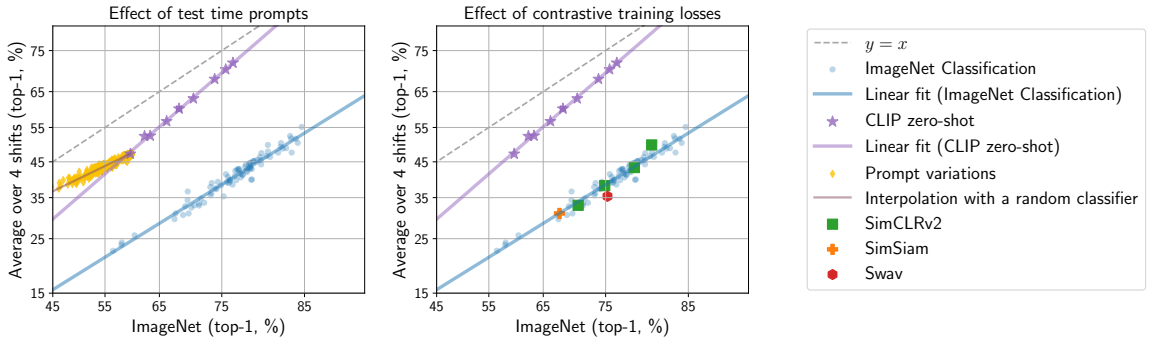

Figure 3.5: Effect of prompting strategies and contrastive objectives on robustness. (Left) On most natural distribution shifts, effect of prompting on effective robustness is similar to that of random interpolation. (Right) Models pre-trained with various contrastive objectives on ImageNet do not achieve the same effective robustness as CLIP models.

Table 3.5: Zero-shot performance for select baselines in the *filtering* track. On all scales, filtering strategies lead to better performance than using the entire, unfiltered pool. The intersection between imaged-based and CLIP score strategies performs well on most tasks and scales. For all metrics, higher is better. ∩ denotes the intersection of filtering strategies.

Scale	Filtering strategy	Dataset size	Samples seen	ImageNet	ImageNet dist. shifts	VTAB	Retrieval	Average over 38 datasets
small	No filtering	12.8M	12.8M	0.025	0.033	0.145	0.114	0.132
	Basic filtering	3M	12.8M	0.038	0.043	0.150	0.118	0.142
	Text-based	3.2M	12.8M	0.046	0.052	0.169	0.125	0.157
	Image-based	3M	12.8M	0.043	0.047	0.178	0.121	0.159
	LAION-2B filtering	1.3M	12.8M	0.031	0.040	0.136	0.092	0.133
	CLIP score (L/14 30%)	3.8M	12.8M	0.051	0.055	0.190	0.119	0.173
	Image-based ∩ CLIP score (L/14 30%)	1.4M	12.8M	0.039	0.045	0.162	0.094	0.144
medium	No filtering	128M	128M	0.176	0.152	0.259	0.219	0.258
	Basic filtering	30M	128M	0.226	0.193	0.284	0.251	0.285
	Text-based	31M	128M	0.255	0.215	0.328	0.249	0.307
	Image-based	29M	128M	0.268	0.213	0.319	0.256	0.312
	LAION-2B filtering	13M	128M	0.230	0.198	0.307	0.233	0.292
	CLIP score (L/14 30%)	38M	128M	0.273	0.230	0.338	0.251	0.328
	Image-based ∩ CLIP score (L/14 30%)	14M	128M	0.297	0.239	0.346	0.231	0.328
large	No filtering	1.28B	1.28B	0.459	0.378	0.426	0.419	0.437
	Basic filtering	298M	1.28B	0.516	0.423	0.446	0.480	0.458
	Text-based	317M	1.28B	0.561	0.465	0.465	0.352	0.466
	Image-based	293M	1.28B	0.572	0.454	0.483	0.479	0.476
	LAION-2B filtering	130M	1.28B	0.553	0.453	0.510	0.495	0.501
	CLIP score (L/14 30%)	384M	1.28B	0.578	0.474	0.538	0.466	0.529
	Image-based ∩ CLIP score (L/14 30%)	140M	1.28B	0.631	0.508	0.546	0.498	0.537
xlarge	No filtering	12.8B	12.8B	0.723	0.612	0.611	0.569	0.621
	LAION-2B filtering	1.3B	12.8B	0.755	0.637	0.624	0.620	0.636
	CLIP score (L/14 30%)	3.8B	12.8B	0.764	0.655	0.643	0.588	0.650
	Image-based ∩ CLIP score (L/14 30%)	1.4B	12.8B	0.792	0.679	0.652	0.608	0.663

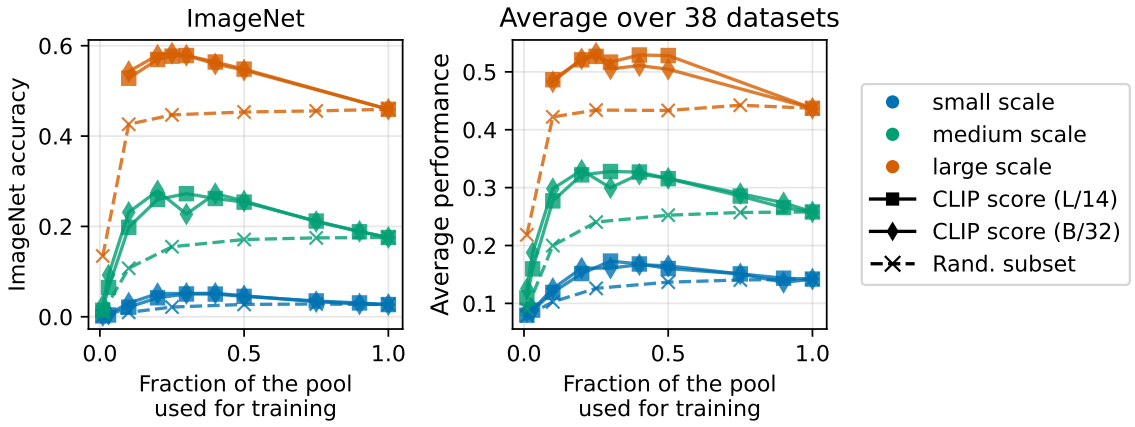

Figure 3.6: Performance of random subsets (dotted line) and CLIP score filtering (solid line) when varying the subset size. When taking random subsets, larger subsets are always better. For CLIP score filtering, subsets with intermediate size perform best.

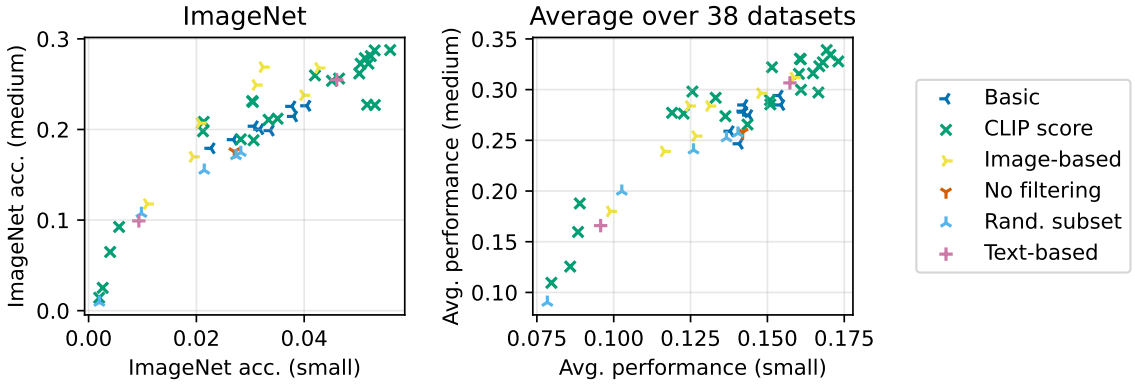

Figure 3.7: Correlation between `small` and `medium` scale baselines. Smaller scales can serve as useful guides for larger scales.

Chapter 4

EDITING MODELS

Despite their versatility, large multimodal models can still perform poorly when used in a zero-shot setting (i.e., without extra training on labeled data). Even when zero-shot models achieve good performance, they are often still worse than models trained or fine-tuned on specific downstream tasks (Radford et al., 2021; Wortsman et al., 2022b; Ilharco et al., 2022). A common practice to address this gap is to fine-tune the zero-shot models on the task of interest. However, fine-tuning often leads to catastrophic forgetting (McCloskey and Cohen, 1989; Thrun, 1998; French, 1999), making the fine-tuned model perform poorly on tasks where the original model performed well (Andreassen et al., 2021; Wortsman et al., 2022b).

This chapter introduces simple techniques that increase accuracy on new tasks without overspecialization. We show that averaging the parameters of zero-shot and fine-tuned models is highly effective at maintaining the best of both worlds: the versatility and robustness of the zero-shot model and the high accuracy on the target task of the fine-tuned model (Wortsman et al., 2022b; Ilharco et al., 2022). Thus, practitioners can *expand* the set of tasks where a LMM has accuracy, for a small fraction of the compute needed for pre-training. This idea of averaging the parameters of different models is further extended in our work model soups (Wortsman et al., 2022a), where we average the parameters of multiple models from a hyper-parameter search. When applying our method to a BASIC-L model (Pham et al., 2021a), the resulting model set a new state-of-the-art on ImageNet, achieving a top-1 accuracy of 90.98%. Since weight averaging results in a single model of the same size, the techniques we propose incur no extra inference cost.

Finally, we introduce task arithmetic, a modular framework of steering the behavior of trained models, designed to effectively add or delete capabilities by directly operating in the weight space (Ilharco et al., 2023). This framework is centered around task vectors,

which are directions in the weight space of a model that steer it towards or away from certain behaviors. We show that it is possible to perform arithmetic operations with these task vectors, which in turn steer the model accordingly. Learning a new task can be done by adding a task vector, and forgetting a task (for example, generating toxic content), by subtracting another. Overall, our work shows that steering models with task arithmetic is effective for various models, tasks and use cases (Ilharco et al., 2023).

4.1 Robust Fine-tuning of Zero-shot Models

In a concrete application, a zero-shot model can be fine-tuned on extra application-specific data, which often yields large performance gains on the target distribution. However, fine-tuning typically comes at the cost of robustness: across several natural distribution shifts, the accuracy of their fine-tuned models is lower than that of the original zero-shot model (Radford et al., 2021; Pham et al., 2021a). This leads to a natural question:

Can zero-shot models be fine-tuned without reducing accuracy under distribution shift?

As pretrained models are becoming a cornerstone of machine learning, techniques for finetuning them on downstream applications are increasingly important. Indeed, the question of robustly finetuning pretrained models has recently also been raised as an open problem by several authors (Andreassen et al., 2021; Bommasani et al., 2021; Radford et al., 2021; Pham et al., 2021b). Andreassen et al. (2021) explored several fine-tuning approaches but found that none yielded models with improved robustness at high accuracy. Furthermore, Taori et al. (2020) demonstrated that no current algorithmic robustness interventions provide consistent gains across the distribution shifts where zero-shot models excel.

In this section, we conduct an empirical investigation to understand and improve fine-tuning of zero-shot models from a distributional robustness perspective. We begin by measuring how different fine-tuning approaches (last-layer vs. end-to-end fine-tuning, hyperparameter changes, etc.) affect the accuracy under distribution shift of the resulting fine-tuned models. Our empirical analysis uncovers two key issues in the standard fine-tuning process. First, the robustness of fine-tuned models varies substantially under even small changes in hyperparameters, but the best hyperparameters cannot be inferred from accuracy on the

target distribution alone. Second, more aggressive fine-tuning (e.g., using a larger learning rate) yields larger accuracy improvements on the target distribution, but can also reduce accuracy under distribution shift by a large amount.

Motivated by the above concerns, we propose a robust way of fine-tuning zero-shot models that addresses the aforementioned trade-off and achieves the best of both worlds: increased performance under distribution shift while maintaining or even improving accuracy on the target distribution relative to standard fine-tuning. In addition, our method simplifies the choice of hyperparameters in the fine-tuning process.

Our method (Figure 4.1) has two steps: first, we fine-tune the zero-shot model on the target distribution. Second, we combine the original zero-shot and fine-tuned models by linearly interpolating between their weights, which we refer to as weight-space ensembling. Interpolating model parameters is a classical idea in convex optimization dating back decades (e.g., see Ruppert (1988); Polyak (1990)). Here, we empirically study model interpolation for non-convex models from the perspective of distributional robustness. Interestingly, linear interpolation in weight-space still succeeds despite the non-linearity in the activation functions of the neural networks.

Weight-space ensembles for fine-tuning (WiSE-FT) substantially improve accuracy under distribution shift compared to prior work while maintaining high performance on the target distribution. Concretely, on ImageNet (Deng et al., 2009) and five of the natural distribution shifts studied by Radford et al. (2021), WiSE-FT applied to standard end-to-end fine-tuning improves accuracy under distribution shift by 4 to 6 percentage points (pp) over prior work while maintaining or improving the ImageNet accuracy of the fine-tuned CLIP model. Relative to the zero-shot model, WiSE-FT improves accuracy under distribution shift by 1 to 9 pp. Moreover, WiSE-FT improves over a range of alternative approaches such as regularization and evaluating at various points throughout fine-tuning. These robustness gains come at no additional computational cost during fine-tuning or inference.

While our investigation centers around CLIP, we observe similar trends for other zero-shot models including ALIGN (Jia et al., 2021), BASIC (Pham et al., 2021b), and a ViT model pre-trained on JFT (Dosovitskiy et al., 2021). For instance, WiSE-FT improves the ImageNet accuracy of a fine-tuned BASIC-L model by 0.4 pp, while improving average

accuracy under distribution shift by 2 to 11 pp.

To understand the robustness gains of WiSE-FT, we first study WiSE-FT when fine-tuning a linear classifier (last layer) as it is more amenable to analysis. In this linear case, our procedure is equivalent to ensembling the outputs of two models, and experiments point towards the complementarity of model predictions as a key property. For end-to-end fine-tuning, we connect our observations to earlier work on the phenomenology of deep learning. Neyshabur et al. (2020) found that end-to-end fine-tuning the same model twice yielded two different solutions that were connected via a linear path in weight-space along which error remains low, known as linear mode connectivity (Frankle et al., 2020; Nagarajan and Kolter, 2019). Our observations suggest a similar phenomenon along the path generated by WiSE-FT, but the exact shape of the loss landscape and connection between error on the target and shifted distributions are still open problems.

In addition to the aforementioned ImageNet distribution shifts, WiSE-FT consistently improves robustness on a diverse set of six additional distribution shifts including: (i) geographic shifts in satellite imagery and wildlife recognition (WILDS-FMoW, WILDS-iWildCam) (Koh et al., 2021; Christie et al., 2018; Beery et al., 2021), (ii) reproductions of the popular image classification dataset CIFAR-10 with a distribution shift (CIFAR-10.1 and CIFAR-10.2) (Recht et al., 2019b; Lu et al., 2020), and (iii) datasets with distribution shift induced by temporal perturbations in videos (ImageNet-Vid-Robust and YTBB-Robust) (Shankar et al., 2019). Beyond the robustness perspective, WiSE-FT also improves accuracy compared to standard fine-tuning, reducing the relative error rate by 4-49% on a range of seven datasets: ImageNet, CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), Describable Textures (Cimpoi et al., 2014), Food-101 (Bossard et al., 2014), SUN397 (Xiao et al., 2016), and Stanford Cars (Krause et al., 2013). Even when fine-tuning data is scarce, reflecting many application scenarios, we find that WiSE-FT improves performance.

Overall, WiSE-FT is simple, universally applicable in the problems we studied, and can be implemented in a few lines of code. Hence we encourage its adoption for fine-tuning zero-shot models.

4.1.1 Weight-space ensembles for fine-tuning

This section describes and motivates our proposed method, WiSE-FT, which consists of two simple steps. First, we fine-tune the zero-shot model on application-specific data. Second, we combine the original zero-shot and fine-tuned models by linearly interpolating between their weights, also referred to as weight-space ensembling. WiSE-FT can be implemented in a few lines of PyTorch.

The zero-shot model excels under distribution shift while standard fine-tuning achieves high accuracy on the reference distribution. Our motivation is to combine these two models into one that achieves the best of both worlds. Weight-space ensembles are a natural choice as they ensemble without extra computational cost. Moreover, previous work has suggested that interpolation in weight space may improve performance when models share part of their optimization trajectory (Izmailov et al., 2018; Neyshabur et al., 2020).

Step 1: Standard fine-tuning. Let $\mathcal{S}_{\text{ref}}^{\text{tr}}$ denote the dataset used for fine-tuning and g denote the image encoder used by CLIP. We are now explicit in writing $g(x, \mathbf{V}_{\text{enc}})$ where x is an input image and $\mathbf{V}_{\text{enc}}$ are the parameters of the encoder g. Standard fine-tuning considers the model $f(x, \theta) = g\left(x, \mathbf{V}_{\text{enc}}\right)^{\top} \mathbf{W}_{\text{classifier}}$ where $\mathbf{W}_{\text{classifier}} \in \mathbb{R}^{d \times k}$ is the classification head and $\theta = [\mathbf{V}_{\text{enc}}, \mathbf{W}_{\text{classifier}}]$ are the parameters of f. We then solve $\arg\min_{\theta} \left\{ \sum_{(x_i, y_i) \in \mathcal{S}_{\text{ref}}^{\text{tr}}} \ell(f(x_i, \theta),\ y_i) + \lambda R(\theta) \right\}$ where ℓ is the cross-entropy loss and R is a regularization term (e.g., weight decay). We consider the two most common variants of fine-tuning: end-to-end, where all values of θ are modified, and fine-tuning only a linear classifier, where $\mathbf{V}_{\text{enc}}$ is fixed at the value learned during pre-training.

Step 2: Weight-space ensembling. For a *mixing coefficient* $\alpha \in [0, 1]$, we consider the *weight-space ensemble* between the zero-shot model with parameters θ_0 and the model obtained via standard fine-tuning with parameters θ_1. The predictions of the weight-space ensemble wse are given by

$$\mathsf{wse}(x, \alpha) = f(x, (1 - \alpha) \cdot \theta_0 + \alpha \cdot \theta_1) \ , \tag{4.1}$$

i.e., we use the element-wise weighted average of the zero-shot and fined-tuned parameters. When fine-tuning only the linear classifier, weight-space ensembling is equivalent to the traditional output-space ensemble (Dietterich, 2000; Breiman, 1996; Freund and Schapire, 1997) $(1-\alpha)\cdot f(x,\theta_0)+\alpha\cdot f(x,\theta_1)$ since Equation 4.1 decomposes as $(1-\alpha)\cdot g(x,\mathbf{V}_{\text{enc}})^\top\mathbf{W}_{\text{zero-shot}}+\alpha\cdot g(x,\mathbf{V}_{\text{enc}})^\top\mathbf{W}_{\text{classifier}}$.

As neural networks are non-linear with respect to their parameters, ensembling all layers—as we do when end-to-end fine-tuning—typically fails, achieving no better accuracy than a randomly initialized neural network (Frankle et al., 2020). However, as similarly observed by previous work where part of the optimization trajectory is shared (Izmailov et al., 2018; Frankle et al., 2020; Neyshabur et al., 2020), we find that the zero-shot and fine-tuned models are connected by a linear path in weight-space along which accuracy remains high (explored further in Section 4.1.3).

Remarkably, as we show in Section 4.1.2, WiSE-FT improves accuracy under distribution shift while maintaining high performance on the reference distribution relative to fine-tuned models. These improvements come without any additional computational cost as a single set of weights is used.

4.1.2 Results

This section presents our key experimental findings. First, we show that WiSE-FT boosts the accuracy of a fine-tuned CLIP model on five ImageNet distribution shifts studied by Radford et al. (2021), while maintaining or improving ImageNet accuracy. Next, we present additional experiments, including more distribution shifts, the effect of hyperparameters, accuracy improvements on the reference distribution, and experiments in the low-data regime. Finally, we demonstrate that our findings are more broadly applicable by exploring WiSE-FT for BASIC (Pham et al., 2021b), ALIGN (Jia et al., 2021), and a ViT-H/14 (Dosovitskiy et al., 2021) model pre-trained on JFT-300M (Sun et al., 2017).

Main results: ImageNet and associated distribution shifts. As illustrated in Figure 4.1, when the mixing coefficient α varies from 0 to 1, $\mathsf{wse}(\cdot,\alpha)$ is able to simultaneously improve accuracy on both the reference and shifted distributions. Table 4.1 presents our

		Distribution shifts					Avg	Avg
	IN (reference)	IN-V2	IN-R	IN-Sketch	ObjectNet	IN-A	shifts	ref., shifts
CLIP `ViT-L/14@336px`								
Zero-shot (Radford et al., 2021)	76.2	70.1	88.9	60.2	70.0	77.2	73.3	74.8
Fine-tuned LC (Radford et al., 2021)	85.4	75.9	84.2	57.4	66.2	75.3	71.8	78.6
Zero-shot (PyTorch)	76.6	70.5	89.0	60.9	69.1	77.7	73.4	75.0
Fine-tuned LC (ours)	85.2	75.8	85.3	58.7	67.2	76.1	72.6	78.9
Fine-tuned E2E (ours)	86.2	76.8	79.8	57.9	63.3	65.4	68.6	77.4
WiSE-FT (ours)								
LC, α=0.5	83.7	76.3	89.6	63.0	70.7	79.7	75.9	79.8
LC, optimal α	85.3	76.9	89.8	63.0	70.7	79.7	75.9	80.2
E2E, α=0.5	86.8	**79.5**	89.4	64.7	71.1	79.9	76.9	81.8
E2E, optimal α	**87.1**	**79.5**	**90.3**	**65.0**	**72.1**	**81.0**	**77.4**	**81.9**

Table 4.1: Accuracy of various methods on ImageNet and derived distribution shifts for CLIP `ViT-L/14@336px` (Radford et al., 2021). E2E: end-to-end; LC: linear classifier. *Avg shifts* displays the mean performance among the five distribution shifts, while *Avg reference, shifts* shows the average of ImageNet (reference) and Avg shifts. For optimal α, we choose the single mixing coefficient that maximizes the column.

main results on ImageNet and five derived distribution shifts. WiSE-FT (end-to-end, α=0.5) outperforms numerous strong models in both average accuracy under distribution shift and the average accuracy on the reference and shifted distributions. While future work may lead to more sophisticated strategies for choosing the mixing coefficient α, α=0.5 yields close to optimal performance across a range of experiments. Hence, we recommend α=0.5 when no domain knowledge is available.

Robustness on additional distribution shifts. Beyond the five distribution shifts derived from ImageNet, WiSE-FT consistently improves robustness on a diverse set of further distributions shifts including geographic shifts in satellite imagery and wildlife recognition (WILDS-FMoW (Koh et al., 2021; Christie et al., 2018), WILDS-iWildCam (Koh et al., 2021; Beery et al., 2021)), reproductions of the popular image classification dataset CIFAR-10 (Krizhevsky et al., 2009) with a distribution shift (CIFAR-10.1 (Recht et al., 2019b) and

CIFAR-10.2 (Lu et al., 2020)), and datasets with distribution shift induced by temporal perturbations in videos (ImageNet-Vid-Robust and YTBB-Robust (Shankar et al., 2020)). Concretely, WiSE-FT (α=0.5) improves performance under distribution shift by 3.5, 6.2, 1.7, 2.1, 9.0 and 23.2 pp relative to the fine-tuned solution while decreasing performance on the reference distribution by at most 0.3 pp (accuracy on the reference distribution often improves). In contrast to the ImageNet distribution shifts, the zero-shot model initially achieves less than 30% accuracy on the WILDS distribution shifts, and WiSE-FT provides improvements regardless.

Hyperparameter variation and alternatives. As illustrated by Figure 4.3, moderate changes in standard hyperparameters such as the learning rate or the number of epochs can substantially affect performance under distribution shift. Moreover, these performance differences cannot be detected reliably from model performance on reference data alone. For instance, while training for 10 epochs with learning rate $3 \cdot 10^{-5}$ and $3 \cdot 10^{-6}$ lead to a small accuracy difference on ImageNet (0.3 pp), accuracy under distribution shift varies by as much as 8 pp.

Furthermore, tuning hyperparameters on ImageNet data can also reduce robustness. For instance, while moving from small to moderate learning rates (10^{-7} to $3 \cdot 10^{-5}$) improves performance on ImageNet by 5 pp, it also deteriorates accuracy under distribution shift by 8 pp.

WiSE-FT addresses this brittleness of hyperparameter tuning: even when using a learning rate $3 \cdot 10^{-5}$ where standard fine-tuning leads to low robustness, applying WiSE-FT removes the trade-off between accuracy on the reference and shifted distributions. The models which can be achieved by varying α are as good or better than those achievable by other hyperparameter configurations. Then, instead of searching over a wide range of hyperparameters, only α needs to be considered. Moreover, evaluating different values of α does not require training new models.

There is no hyperparameter in Figure 4.3 which can be varied to match or exceed the optimal curve produced by WiSE-FT. In our experiments, this frontier is reached only through methods that average model weights, either using WiSE-FT or with a more so-

	ImageNet	CIFAR10	CIFAR100	Cars	DTD	SUN397	Food101
Standard fine-tuning	86.2	98.6	92.2	91.6	81.9	80.7	94.4
WiSE-FT (α=0.5)	86.8 (+0.6)	99.3 (+0.7)	93.3 (+1.1)	93.3 (+1.7)	84.6 (+2.8)	83.2 (+2.5)	96.1 (+1.6)
WiSE-FT (opt. α)	87.1 (+0.9)	99.5 (+0.8)	93.4 (+1.2)	93.6 (+2.0)	85.2 (+3.3)	83.3 (+2.6)	96.2 (+1.8)

Table 4.2: Beyond robustness, WiSE-FT can improve accuracy after fine-tuning on several datasets.

phisticated averaging scheme: keeping an exponential moving average of all model iterates (EMA, Szegedy et al. (2016)).

Accuracy gains on reference distributions. Beyond robustness to distribution shift, Table 4.2 demonstrates that WiSE-FT also improves accuracy after fine-tuning on seven datasets. When fine-tuning end-to-end on ImageNet, CIFAR-10, CIFAR-100, Describable Textures, Food-101, SUN397, and Stanford Cars, WiSE-FT reduces relative error by 4 to 49%. Even though standard fine-tuning directly optimizes for high accuracy on the reference distribution, WiSE-FT achieves better performance.

Beyond CLIP. Figure 4.4 illustrates that WiSE-FT is generally applicable to zero-shot models beyond CLIP, and beyond models pre-trained contrastively with image-text pairs. First, we interpolate between the weights of the zero-shot and fine-tuned BASIC-L model (Pham et al., 2021b), finding that α=0.5 improves average accuracy on five distribution shifts derived from ImageNet by over 7 pp while improving ImageNet accuracy by 0.4 pp relative to the fine-tuned BASIC-L model. As in Pham et al. (2021b), the model is fine-tuned using a contrastive loss and half of the ImageNet training data. WiSE-FT provides improvements on both reference and shifted distributions, despite these experimental differences.

Next, we consider the application of WiSE-FT to a ViT-H/14 model (Dosovitskiy et al., 2021) pre-trained on JFT-300M (Sun et al., 2017), where the zero-shot classifier is constructed by manually identifying a class correspondence. WiSE-FT improves performance

under distribution shift over both the zero-shot and fine-tuned models. When α=0.8, WiSE-FT outperforms the fine-tuned model by 2.2 pp on distribution shifts, while maintaining ImageNet performance within 0.2 pp of the fine-tuned model. This result demonstrates that WiSE-FT can be successfully applied even to models which do not use contrastive image-text pre-training.

Finally, we apply WiSE-FT to the ALIGN model of Jia et al. (2021), which is similar to CLIP but is pre-trained with a different dataset, finding similar trends.

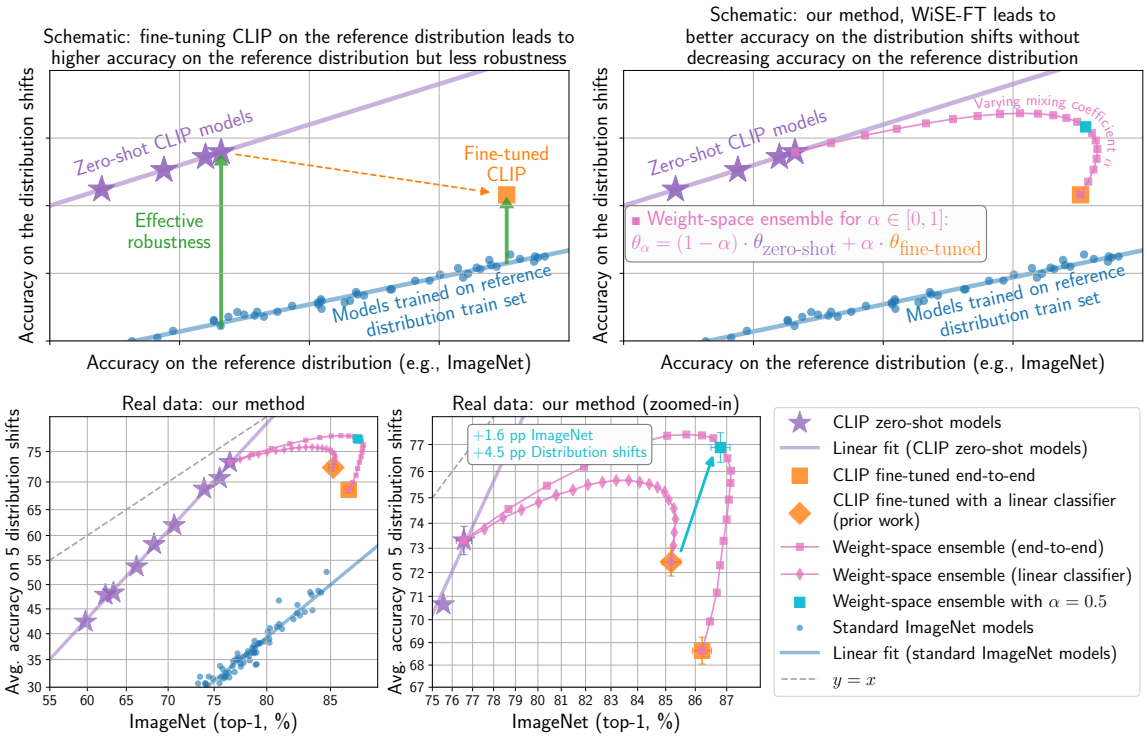

Figure 4.1: **(Top left)** Zero-shot CLIP models exhibit moderate accuracy on the reference distribution (x-axis, the target for fine-tuning) and high effective robustness (accuracy on the distribution shifts beyond the baseline models). In contrast, standard fine-tuning—either end-to-end or with a linear classifier (final layer)—attains higher accuracy on the reference distribution but less effective robustness. **(Top right)** Our method linearly interpolates between the zero-shot and fine-tuned models with a mixing coefficient $\alpha \in [0, 1]$. **(Bottom)** On five distribution shifts derived from ImageNet (ImageNetV2, ImageNet-R, ImageNet Sketch, ObjectNet, and ImageNet-A), WiSE-FT improves average accuracy relative to both the zero-shot and fine-tuned models while maintaining or improving accuracy on ImageNet.

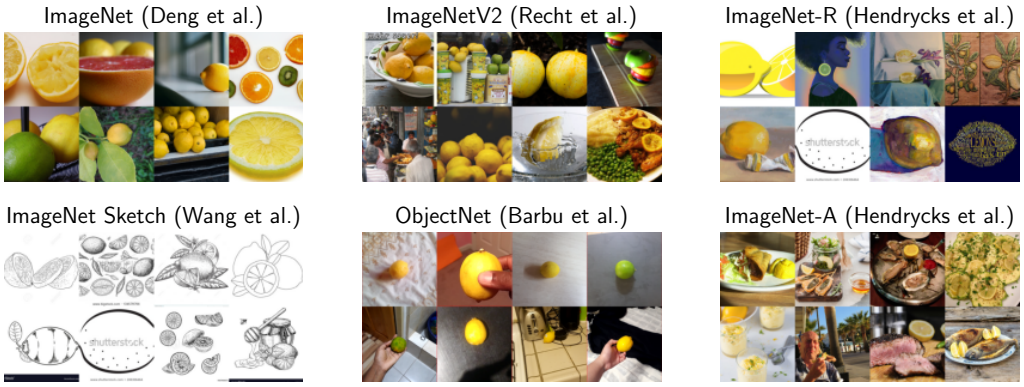

Figure 4.2: Samples of the class *lemon*, from the reference distribution ImageNet (Deng et al., 2009) and the derived distribution shifts considered in our main experiments: ImageNet-V2 (Recht et al., 2019b), ImageNet-R (Hendrycks et al., 2021a), ImageNet Sketch (Wang et al., 2019), ObjectNet (Barbu et al., 2019), and ImageNet-A (Hendrycks et al., 2021b).

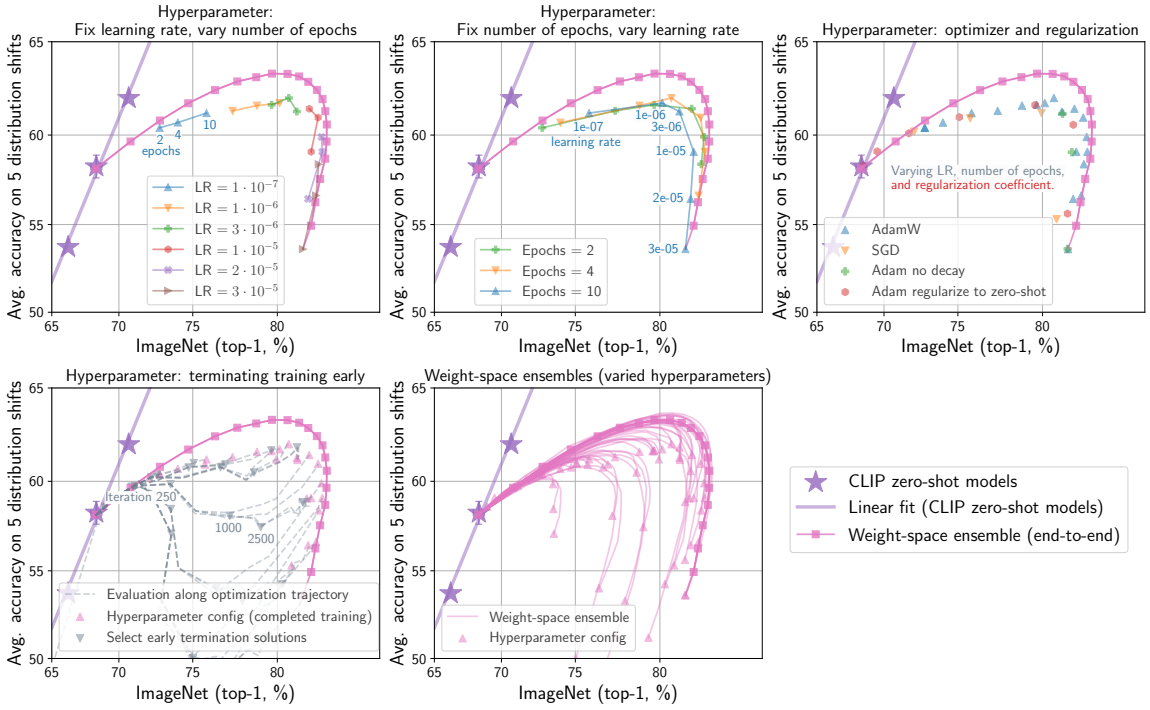

Figure 4.3: The robustness of fine-tuned models varies substantially under even small changes in hyperparameters. Applying WiSE-FT addresses this brittleness and can remove the trade-off between accuracy on the reference and shifted distributions. Results shown for CLIP `ViT-B/16` fine-tuned with cosine-annealing learning rate schedule and all models in the top left and top middle plots are fine-tuned with AdamW (Loshchilov and Hutter, 2019). Moreover, *regularize to zero-shot* appends the regularizer $\lambda\|\theta - \theta_0\|_2^2$ to the fine-tuning objective, where θ_0 are the parameters of the zero-shot model.

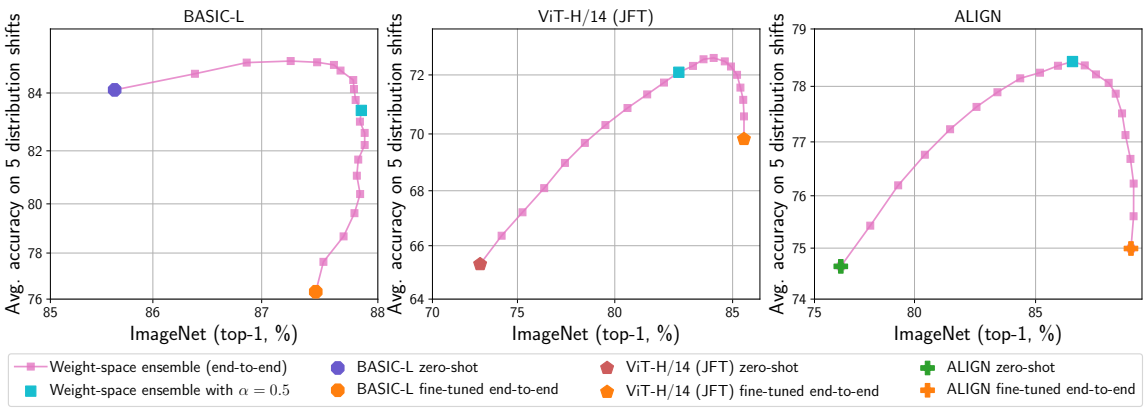

Figure 4.4: WiSE-FT applied to BASIC-L (Pham et al., 2021b), a ViT-H/14 (Dosovitskiy et al., 2021) model pre-trained on JFT-300M (Sun et al., 2017) and ALIGN (Jia et al., 2021).

4.1.3 Discussion

This section further analyzes the empirical phenomena we have observed so far. We begin with the case where only the final linear layer is fine-tuned and predictions from the weight-space ensemble can be factored into the outputs of the zero-shot and fine-tuned model. Next, we connect our observations regarding end-to-end fine-tuning with earlier work on the phenomenology of deep learning.

Zero-shot and fine-tuned models are complementary

In this section, we find that the zero-shot and fine-tuned models have diverse predictions, both on reference and shifted distributions. Moreover, while the fine-tuned models are more confident on the reference distribution, the reverse is true under distribution shift.

Zero-shot and fine-tuned models are diverse. In certain cases, ensemble accuracy is correlated with diversity among the constituents (Kuncheva and Whitaker, 2003; Gontijo-Lopes et al., 2021). If two models make coincident mistakes, so will their ensemble, and no benefit will be gained from combining them. Here, we explore two measures of diversity: *prediction diversity,* which measures the fraction of examples for which two classifiers disagree but one is correct; and *Centered Kernel Alignment Complement,* the complement of CKA (Kornblith et al., 2019a). In Figure 4.5 (left), we show that the zero-shot and fine-tuned models are diverse both on the reference and shifted distributions, despite sharing the same backbone. As a point of comparison, we include avg. diversity measures between two linear classifiers fine-tuned with random splits on half of ImageNet,[1] denoted in orange in Figure 4.5.

Models are more confident where they excel. In order for the ensemble model to be effective, it should leverage each model's expertise based on which distribution the data is from. Here, we empirically show that this occurs on a number of datasets we consider.

[1]Two linear classifiers fine-tuned on the same data converge to similar solutions, resulting in negligible diversity. As a stronger baseline, we fine-tune classifiers on different subsets of ImageNet, with half of the data.

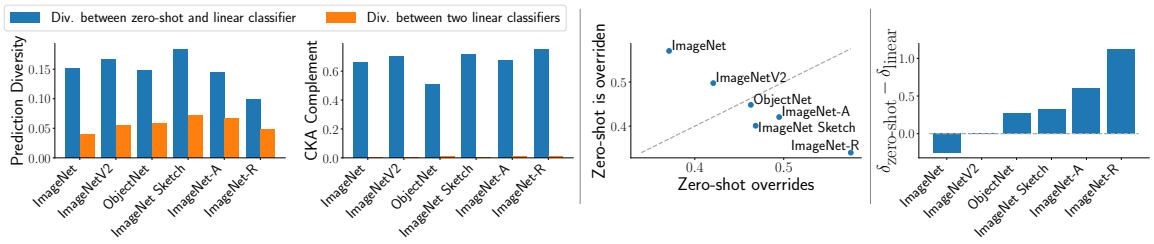

Figure 4.5: **(Left)** Zero-shot and fine-tuned models exhibit diversity in their predictions. **(Middle)** On most distribution shifts, the zero-shot model overrides the linear classifier more than it is overridden. The reverse is true for ImageNet (reference). **(Right)** Similarly, zero-shot models are more confident under distribution shift, while the reverse is true on the reference distribution. The margin δ_f measures the average difference between the largest and second largest unormalized output for classifier f.

First, we examine the cases where the models being ensembled disagree. We say the zero-shot model *overrides* the fine-tuned model if their predictions disagree and the zero-shot prediction matches that of the weight-space ensemble. Similarly, if models disagree and the linear classifier prediction matches the ensemble, we say the zero-shot is *overridden*. Figure 4.5 (middle) shows the fraction of samples where the zero-shot model overrides and is overridden by the fine-tuned linear classifier for α=0.5. Other than ImageNetV2, which was collected to closely reproduce ImageNet, the zero-shot model overrides the linear classifier more than it is overridden on the distribution shifts.

Additionally, we are interested in measuring model confidence. Recall that we are ensembling quantities before a softmax is applied, so we avoid criteria that use probability vectors, e.g., Guo et al. (2017). Instead, we consider the margin δ between the largest and second largest output of each classifier. Figure 4.5 (right) shows that the zero-shot model is more confident in its predictions under distribution shift, while the reverse is true on the reference distribution.

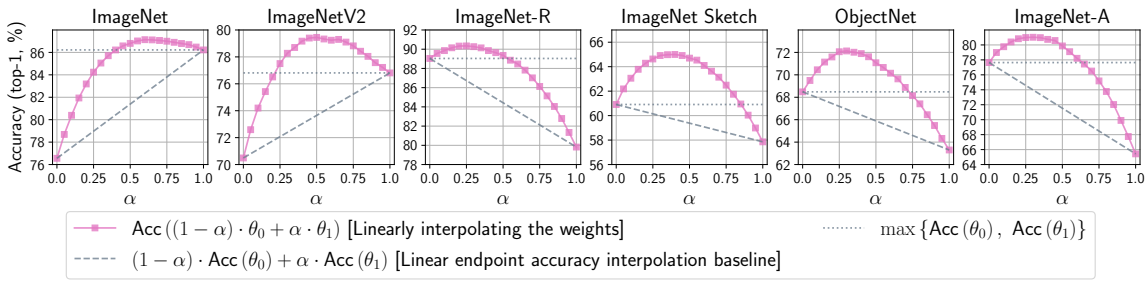

Figure 4.6: On ImageNet and the main distribution shifts we consider, linearly interpolating between the weights of θ_0 and θ_1 exceeds the baseline of linearly interpolating the accuracies of the two models for all α (Observation 1). Moreover, there exists an α for which WiSE-FT outperforms both the zero-shot and fine-tuned models (Observation 2).

An error landscape perspective

We now turn to empirical phenomena we observe when weight-space ensembling *all* layers in the network. Specifically, this section formalizes our observations and details related phenomena. Recall that the weight-space ensemble of θ_0 and θ_1 is given by $f(x, (1-\alpha)\cdot\theta_0 + \alpha\cdot\theta_1)$ (Equation 4.1).

For a distribution $\mathcal{D}$ and model f, let $\mathsf{Acc}_{\mathcal{D},f}(\theta)$ denote the expected accuracy of f evaluated with parameters θ on distribution $\mathcal{D}$.

Observation 1: As illustrated in Figure 4.6, on ImageNet and the five associated distribution shifts we consider

$$\mathsf{Acc}_{\mathcal{D},f}((1-\alpha)\cdot\theta_0 + \alpha\cdot\theta_1) \;\geq\; (1-\alpha)\cdot\mathsf{Acc}_{\mathcal{D},f}(\theta_0) + \alpha\cdot\mathsf{Acc}_{\mathcal{D},f}(\theta_1) \tag{4.2}$$

for all $\alpha \in [0, 1]$.

Note that equation 4.2 uses the baseline of linearly interpolating between the accuracies of the two endpoints, which is always achievable by using weights θ_1 with probability α and using model θ_0 otherwise. In the case where the accuracy of both endpoints are similar, Equation 4.2 is equivalent to the definition of Linear Mode Connectivity of Frankle et al. (2020).

To assist in contextualizing Observation 1, we review related phenomena. Neural net-

works are nonlinear, hence weight-space ensembles only achieve good performance in exceptional cases—interpolating the weights of two networks trained on ImageNet from the same random initialization results in no better accuracy than a random classifier (Frankle et al., 2020). On the simpler MNIST task (LeCun, 1998), linear mode connectivity was observed by Nagarajan and Kolter (2019) between a pair of models trained from the same random initialization. Linear mode connectivity has also been observed for harder tasks such as ImageNet by Frankle et al. (2020); Izmailov et al. (2018) when part of the training trajectory is shared. Finally, Neyshabur et al. (2020) observe linear mode connectivity between two models that are fine-tuned from a shared, pre-trained initialization. In particular, the observations of Neyshabur et al. (2020) may elucidate why weight-space ensembles attain high accuracy in the setting we consider, as they suggest that fine-tuning remains in a region where solutions are connected by a linear path along which error remains low. Instead of considering the weight-space ensemble of two fine-tuned models, we consider the weight-space ensemble of the *pre-trained* and fine-tuned models. This is only possible for a pre-trained model capable of zero-shot inference such as CLIP.

Observation 2: As illustrated by Figure 4.6, on ImageNet and the five associated distribution shifts we consider, weight-space ensembling (end-to-end) may outperform both the zero-shot and fine-tuned models, i.e., there exists an α for which $\mathsf{Acc}_{\mathcal{D},f}\left((1-\alpha)\cdot\theta_0+\alpha\cdot\theta_1\right) \geq \max\left\{\mathsf{Acc}_{\mathcal{D},f}\left(\theta_0\right),\ \mathsf{Acc}_{\mathcal{D},f}\left(\theta_1\right)\right\}$.

We are not the first to observe that when interpolating between models, the accuracy of models along the path may exceed that of either endpoint (Izmailov et al., 2018; Neyshabur et al., 2020; Wortsman et al., 2021). Neyshabur et al. (2020) conjecture that interpolation could produce solutions closer to the true center of a basin. In contrast to Neyshabur et al. (2020), we interpolate between models which observe different data.

4.1.4 Conclusion

WiSE-FT can substantially improve performance under distribution shift with minimal or no loss in accuracy on the target distribution compared to standard fine-tuning. Overall, we view WiSE-FT as a first step towards more sophisticated fine-tuning schemes that we

hope will continue to leverage the robustness of zero-shot models for building more reliable neural networks.

4.2 Model soups

Typically, the fine-tuning process involves two steps. First, fine-tune models with a variety of hyperparameter configurations. Then, select the model which achieves the highest accuracy on the held-out validation set. The remaining models are then discarded.

Selecting a single model and discarding the rest has several downsides. For one, ensembling outputs of many models can outperform the best single model, but only at a high computational cost during inference. For another, fine-tuning a model on downstream tasks can sometimes reduce out-of-distribution performance (Radford et al., 2021; Andreassen et al., 2021; Wortsman et al., 2022b; Pham et al., 2021b), and the best single model on the target distribution may not be the best model on out-of-distribution data.

In our work Wortsman et al. (2022a), we propose a more accurate and robust alternative to the second step of the conventional recipe in the context of fine-tuning a large pre-trained model. Instead of selecting the individual fine-tuned model which achieves the highest accuracy on the held-out validation set, we average the weights of models fine-tuned independently, and refer to the result as a *model soup*. Given the results of the first step—a hyperparameter sweep over fine-tuned models—averaging several of these models to form a model soup requires no additional training and adds no cost at inference time.

Since the loss landscape of neural network training is non-convex with many solutions in different loss basins, it is perhaps surprising that averaging the weights of independently fine-tuned models achieves high performance. However, recent work (Neyshabur et al., 2020) observes that fine-tuned models optimized independently from the same pre-trained initialization lie in the same basin of the error landscape, inspiring our method. Weight averaging along a single training trajectory has previously been shown to improve the performance of models in non-transfer settings (Szegedy et al., 2016; Izmailov et al., 2018). Our approach extends weight averaging to the context of fine-tuning, where we find that it also works across many independent runs with varied hyperparemeter configurations. Our use of a diverse set of fine-tuned models is inspired by Gontijo-Lopes et al. (2021) who

observe that ensembling independent runs trained with different hyperparameters improves performance.

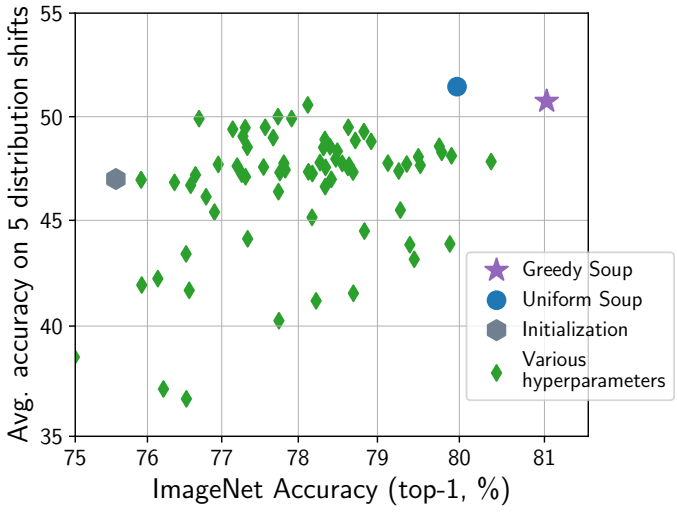

Figure 4.7: *Model soups* improve accuracy over the best individual model when performing a large, random hyperparameter search for fine-tuning a CLIP ViT-B/32 model on ImageNet. The *uniform soup* (blue circle) averages all fine-tuned models (green diamonds) in a random hyperparameter search over learning rate, weight-decay, iterations, data augmentation, mixup, and label smoothing. The *greedy soup* adds models sequentially to the model soup, keeping a model in the soup if accuracy on the held-out validation set does not decrease.

We perform a comprehensive experimental study of fine-tuning to understand the behavior of model soups. For our main results we fine-tune CLIP (Radford et al., 2021) and ALIGN (Jia et al., 2021), which are pre-trained with a contrastive loss on image-text pairs, and a ViT-G model pre-trained on JFT (Zhai et al., 2021). Our results show that model soups often outperform the best individual model on both the in-distribution and natural distribution shift test sets. A model soup composed of ViT-G models achieves 90.94% on ImageNet (Deng et al., 2009), surpassing the previous state of the art of 90.88% attained by the CoAtNet model (Dai et al., 2021) while requiring 25% fewer FLOPs at inference time. In general, model soups can approach the performance of ensembling, with no additional

computational cost or memory relative to a single model during inference. Beyond ImageNet and associated distribution shifts, our results show that model soups are applicable when fine-tuning on tasks from the WILDS (Koh et al., 2021) benchmark, and when fine-tuning transformer models (Vaswani et al., 2017; Devlin et al., 2019a; Raffel et al., 2020c) for text classification.

While the most straightforward approach to making a model soup is to average all the weights uniformly, we find that *greedy soups*, where models are sequentially added to the soup if they improve accuracy on held-out data, outperforms uniform averaging. Greedy soups avoid adding in models which may lie in a different basin of the error landscape, which could happen if, for example, models are fine-tuned with high learning rates.

In addition to empirical observations, we analytically relate the similarity in loss between weight-averaging and logit-ensembling to the flatness of the loss (i.e., its second derivative on a line between models) and confidence of the predictions (expressed via the variance of a logits difference drawn from the weight-average softmax). We empirically validate our approximation on a subset of the models we train and show that it is strongly correlated with the true averaging vs. ensembling performance difference, particularly in the learning rate regimes where soups are effective and models achieve higher accuracy.

4.2.1 Method

This section highlights three recipes for model souping, the *uniform*, *greedy*, and *learned* soup, though the greedy soup is our central method. We consider a neural network $f(x, \theta)$ with input data x and parameters $\theta \in \mathbb{R}^d$. Fine-tuning is analogous to standard neural network training but includes an important distinction: the parameters are initialized to those found via pre-training.

Let $\theta = \mathsf{FineTune}(\theta_0, h)$ denote the parameters obtained by fine-tuning with pre-trained initialization θ_0 and hyperparameter configuration h. The hyperparameter configuration can include the choice of optimizer, data augmentation, training iterations, and a random seed which will determine data order.

For hyperparameter configurations $h_1, ..., h_k$ let $\theta_i = \mathsf{FineTune}(\theta_0, h_i)$. Conventionally,

the parameters θ_j which attain the highest accuracy on a held out validation set are selected, and the remaining parameters are discarded. Instead, *model soups* $f(x, \theta_{\mathcal{S}})$ use an average of θ_i, i.e., $\theta_{\mathcal{S}} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \theta_i$ where $\mathcal{S} \subseteq \{1, ..., k\}$. The *uniform soup* is constructed by averaging *all* fine-tuned models θ_i and so $\mathcal{S} = \{1, ..., n\}$.

There are settings in which a hyperparameter configuration can produce a model with low accuracy that results in a low accuracy uniform soup. This issue can be circumvented with a *greedy soup*. The greedy soup is constructed by sequentially adding each model as a potential ingredient in the soup, and only keeping the model in the soup if performance on a held out validation set (disjoint from the training and test sets) improves. Before running this procedure we sort the models in decreasing order of validation set accuracy, and so the greedy soup can be no worse than the best individual model on the held-out validation set. We also explore a more advanced *learned soup* recipe that optimizes model interpolation weights by gradient-based minibatch optimization. This procedure requires simultaneously loading all models in memory which currently hinders its use with large networks. More details are available at (Wortsman et al., 2022a).

4.2.2 Experiments

This section presents our key experimental findings. We begin with experimental setup then provide intuition for model soups by examining error landscape visualizations. Next we present our main results, using model soups as an alternative to selecting the best performing individual model.

Experimental setup

Our experiments explore the application of model soups when fine-tuning various models. The primary models we fine-tune are the CLIP (Radford et al., 2021), ALIGN (Jia et al., 2021), and BASIC (Pham et al., 2021b) models pre-trained with contrastive supervision from image-text pairs, a ViT-G/14 model pre-trained on JFT-3B (Zhai et al., 2021), and transformer models for text classification (Devlin et al., 2019a; Raffel et al., 2020a). Unless otherwise mentioned, experiments use the CLIP ViT-B/32 model. Fine-tuning is performed

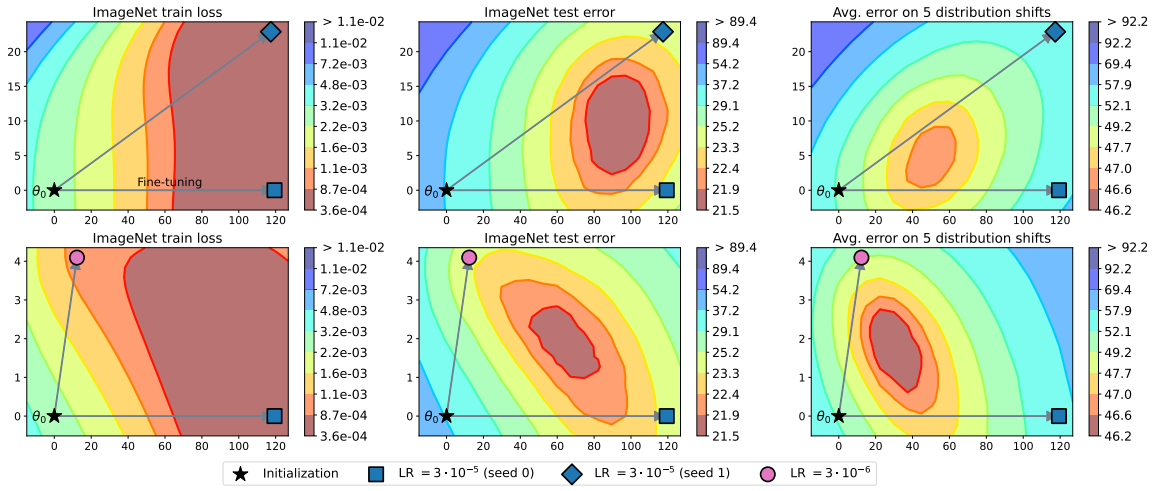

Figure 4.8: The solution with the highest accuracy is often not a fine-tuned model but rather lies between fine-tuned models. This figure shows loss and error on a two dimensional slice of the loss and error landscapes. We use the zero-shot initialization θ_0 and fine-tune twice (illustrated by the gray arrows), independently, to obtain solutions θ_1 and θ_2. As in Garipov et al. (2018), we obtain an orthonormal basis u_1, u_2 for the plane spanned by these models, and the x and y-axis show movement in parameter space in these directions, respectively.

end-to-end (all parameters are modified) which typically results in better accuracy than training only the final linear layer (Kornblith et al., 2019b; Agrawal et al., 2014; Chatfield et al., 2014; Azizpour et al., 2015).

We consider two different methods for initializing the final linear layer before fine-tuning. The first method initializes the model from a linear probe (LP), as described in Kumar et al. (2022b), and we refer to this method as LP initialization. The second method uses the zero-shot initialization, e.g., using the classifier produced by the text tower of CLIP or ALIGN as the initialization. Both methods for initializing the model produce similar trends when applicable, and unless otherwise stated we use the LP initialization.

For the ensemble baselines (Dietterich, 2000; Lakshminarayanan et al., 2017) we ensemble the logits (unormalized outputs) of models as in Gontijo-Lopes et al. (2021). Fine-tuning uses a supervised cross-entropy loss and, unless otherwise mentioned, is conducted

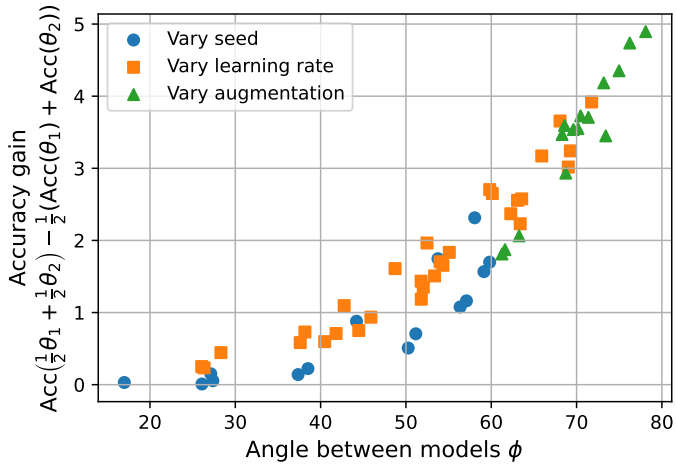

Figure 4.9: The advantage of averaging solutions (y-axis) is correlated with the angle ϕ between between solutions, while varying hyperparameter configurations between pairs enables a larger ϕ. Each point corresponds to a pair of models θ_1, θ_2 that are fine-tuned independently from a shared initialization θ_0 with different hyperparameter configurations. The angle ϕ between between solutions refers to the angle between $\theta_1 - \theta_0$ and $\theta_2 - \theta_0$ (i.e., the initialization is treated as the origin). Accuracy is averaged over ImageNet and the five distribution shifts described in Section 4.2.2.

on ImageNet (Deng et al., 2009). When fine-tuning on ImageNet we also evaluate on the five natural distribution shifts: ImageNetV2 (Recht et al., 2019b), ImageNet-R (Hendrycks et al., 2021a), ImageNet-Sketch (Wang et al., 2019), ObjectNet (Barbu et al., 2019), and ImageNet-A (Hendrycks et al., 2021b). We often report results averaged over these five distribution shifts. Since the official ImageNet validation set is typically used as the test set, we use roughly 2% of the ImageNet training set as a held-out validation set for constructing greedy soups.

4.2.3 Intuition and motivation

Error landscape visualizations. To provide intuition, we visualize a two dimensional slice of the training loss and test error landscape when fine-tuning CLIP on ImageNet. In these experiments, we use the zero-shot initialization $\theta_0 \in \mathbb{R}^d$ and fine-tune twice, indepen-

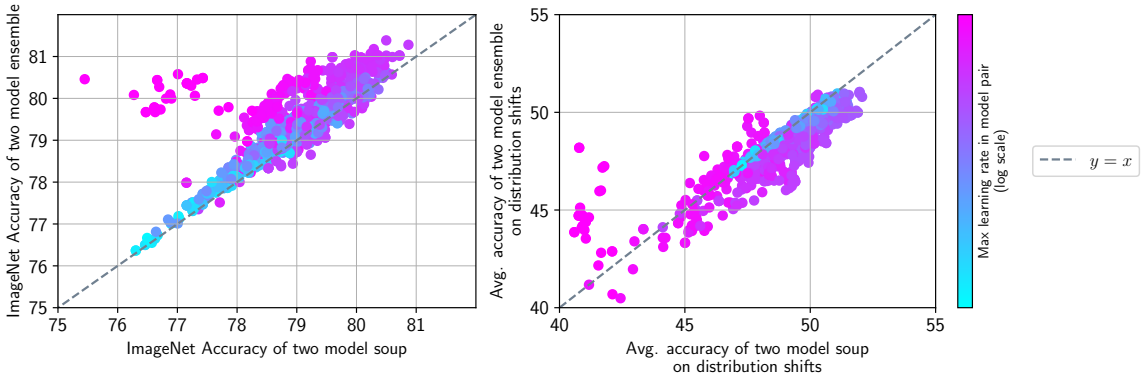

Figure 4.10: Ensemble performance is correlated with model soup performance. Each point on the scatter plot is a model pair with different hyperparameters. The x-axis is the accuracy when the weights of the two models are averaged (i.e., the two model soup) while the y-axis is the accuracy of the two model ensemble. Ensembles often perform slightly better than soups on ImageNet (left) while the reverse is true on the distribution shifts (right).

dently, to produce solutions θ_1 and θ_2. The points θ_0, θ_1 and θ_2 define a plane in parameter space, and we evaluate the ImageNet train loss, ImageNet test error, and the test error on the five aforementioned distribution shifts on this plane. The results are illustrated in Figure 4.8 where the zero-shot initialization (θ_0) is shown as a star and a solution fine-tuned with learning rate $3 \cdot 10^{-5}$ (θ_1) is shown as a blue square. For θ_2 we either use the same learning rate as θ_1 (but vary the random seed) or learning rate $3 \cdot 10^{-6}$. For both the in-distribution and out-of-distribution test sets, the loss/error contours are basin-shaped, and none of the three points is optimal.

These results suggest that (1) interpolating the weights of two fine-tuned solutions can improve accuracy compared to individual models and (2) more uncorrelated solutions—models that form an angle[2] closer to 90 degrees—may lead to higher accuracy on the linear interpolation path.

To investigate the correlation between accuracy improvement and angle, we consider a

[2]In particular, the angle ϕ between $\theta_1 - \theta_0$ and $\theta_2 - \theta_0$, i.e., the angle between the arrows shown in Figure 4.8.

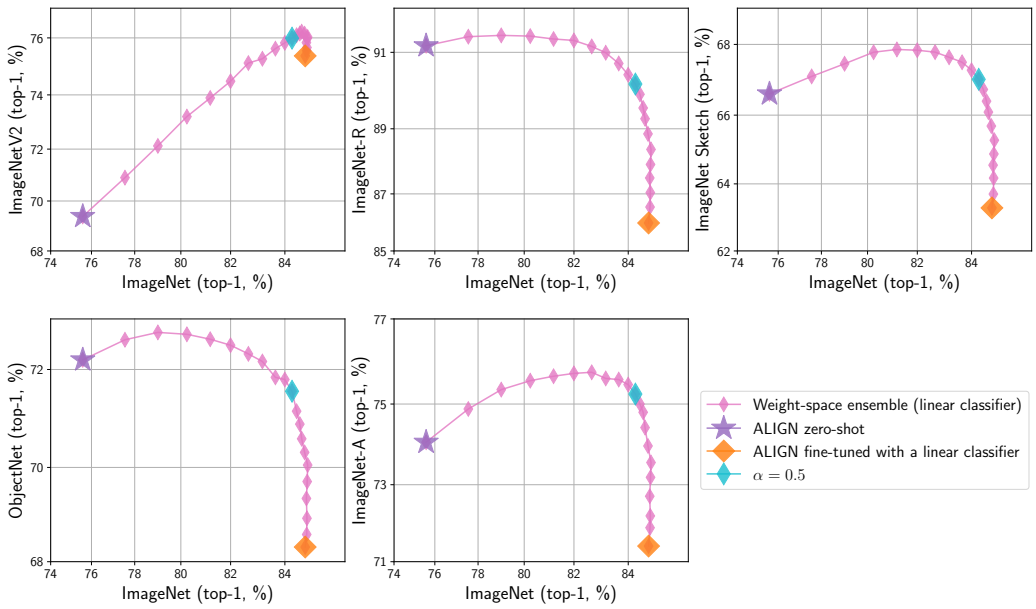

Figure 4.11: *Model soups* improve accuracy when fine-tuning ALIGN.

series of models trained with different seeds, learning rates, and data augmentation. For each pair θ_1, θ_2, we compare the accuracy of their average with the average of their accuracies, $\mathsf{Acc}\left(\frac{1}{2}\theta_1 + \frac{1}{2}\theta_2\right) - \frac{1}{2}\left(\mathsf{Acc}\left(\theta_1\right) + \mathsf{Acc}\left(\theta_2\right)\right)$, which we refer to as the interpolation advantage. Figure 4.9 illustrates the results, in which we observe that the interpolation advantage is correlated with the angle ϕ and that varying the learning rate, seed, or data augmentation can produce solutions which are more orthogonal.

Ensemble comparison. Figure 4.10 observes that ensemble performance is correlated with soup performance for moderate and small learning rates. We consider pairs of models selected at random from individual solutions, and find that the maximum learning rate of the models in the pair is indicative of the ensemble accuracy, soup accuracy, and their relation: When learning rate is small, ensemble accuracy and soup accuracy are similar, but both are suboptimal. For moderate learning rate values, ensemble accuracy and soup accuracy are both high. For high learning rate values, ensemble performance exceeds soup performance, but ensembles/soups with moderate learning rates perform better. Overall, ensembles achieve higher accuracy on ImageNet while the reverse is true on the distribution

shifts.

One dimensional hyperparameter grids. Finally, we ask the question: for a one dimensional grid of hyperparameters $\{h_a, ..., h_b\}$, how does averaging the models fine-tuned with hyperparameter configurations h_a and h_b corresponding to the endpoints compare with picking the best individual model fine-tuned with hyperparameter configuration $h \in \{h_a, ..., h_b\}$? The hyperparameters we vary are optimizer, augmentation, and learning rate. For the majority of grid searches, the average of the endpoints outperforms the best individual model in the grid.

4.2.4 Model soups

With the gains of averaging two fine-tuned models in mind, we turn our attention to averaging *many* models with different hyperparameters: this section presents our main results, which show that averaging fine-tuned models can be used as an alternative to the conventional procedure of selecting the single model which performs best on the held-out validation set. We explore CLIP (Radford et al., 2021) and ALIGN (Jia et al., 2021) fine-tuned on ImageNet (Deng et al., 2009) (Section 4.2.4), ViT-G pre-trained on JFT-3B (Zhai et al., 2021) and fine-tuned on ImageNet (Section 4.2.4), and transformer models fine-tuned on text classification tasks (Section 4.2.4).

Fine-tuning CLIP and ALIGN

We begin our study of model soups by considering two-pretrained models, CLIP ViT-B/32 and ALIGN EfficientNet-L2, and performing a hyperparameter sweep for the fine-tuning each model on ImageNet. For CLIP we use a random hyperparameter search over learning rate, weight decay, training epochs, label smoothing, and data augmentation, obtaining 72 fine-tuned models. For ALIGN we use a grid search over learning rate, data augmentation, and mixup, obtaining 12 fine-tuned models. To form our greedy soups, we sort models in order of decreasing accuracy on the held-out validation set. For both CLIP and ALIGN, the greedy soup selects 5 models. Figure 4.7 and 4.11 show the performance of the resulting models and their uniform and greedy soups for CLIP and ALIGN. The greedy soup improves

Table 4.3: Ablation on multiple methods and their variants when when fine-tuning CLIP ViT-B/32 with the random hyperparameter search described in Section 4.2.4. For "Greedy soup (random order)", we try three random model orders when running the greedy soup procedure (by default, models are sorted by decreasing held-out val accuracy). The *best* in *best individual model* refers to ImageNet accuracy.

	ImageNet	Dist. shifts
Best individual model	80.38	47.83
Second best model	79.89	43.87
Uniform soup	79.97	51.45
Greedy soup	81.03	50.75
Greedy soup (random order)	80.79 (0.05)	51.30 (0.16)
Learned soup	80.89	51.07
Learned soup (by layer)	81.37	50.87
Ensemble	81.19	50.77
Greedy ensemble	81.90	49.44

on over the best model in the hyperparameter sweep by 0.7 and 0.5 percentage points, respectively.

Furthermore, we show that, for essentially any number of models, the greedy soup outperforms the best single model on both the ImageNet and the out-of-distribution test sets. We consider an additional setting where we prepare a sequence of soups by sequentially adding CLIP models from the hyperparameter sweep in random order. We investigate the performance of the uniform and greedy soup, as well as the best single model so far and a logit ensemble, as a function of the number of models considered. The greedy soup is better than the uniform soup on ImageNet and comparable to it out-of-distribution. The logit ensemble is better than the greedy soup on ImageNet, but worse out-of-distribution.

Table 4.3 lists the performance of the CLIP soups and baselines. To further establish the generality of the model soup, we replicate the CLIP hyperparameter sweep experiment on two image classification tasks from WILDS (Koh et al., 2021), namely FMoW (Christie et al., 2018) and iWildCam (Beery et al., 2021).

Table 4.4: Greedy soup improves over the best individual models obtained in a hyperparameter sweep for ViT-G/14 pre-trained on JFT-3B and fine-tuned on ImageNet, both in- and out-of-distribution. Accuracy numbers not significantly different from the best are boldfaced. Statistical comparisons are performed using an exact McNemar test or permutation test at $\alpha = 0.05$. Avg shift accuracy of the best model on each test set is the best average accuracy of any individual model.

	ImageNet			Distribution shifts					
Method	Top-1	ReaL	Multilabel	IN-V2	IN-R	IN-Sketch	ObjectNet	IN-A	Avg shifts
ViT/G-14 (Zhai et al., 2021)	90.45	90.81	–	83.33	–	–	70.53	–	–
CoAtNet-7 (Dai et al., 2021)	90.88	–	–	–	–	–	–	–	–
Our models/evaluations based on ViT-G/14:									
ViT/G-14 (Zhai et al., 2021) (reevaluated)	90.47	90.86	96.89	83.39	94.38	72.37	71.16	89.00	82.06
Best model on held out val set	90.72	91.04	96.94	83.76	95.04	73.16	78.20	91.75	84.38
Best model on each test set (oracle)	90.78	**91.78**	**97.29**	**84.31**	95.04	73.73	**79.03**	92.16	84.68
Greedy ensemble	**90.93**	91.29	**97.23**	**84.14**	94.85	73.07	77.87	91.69	84.33
Greedy soup	**90.94**	91.20	**97.17**	**84.22**	**95.46**	**74.23**	78.52	**92.67**	**85.02**

We report several additional variants and baselines for the experiment described above, and highlight a few interesting takeaways: (1) The greedy soup outperforms the best individual model—with no extra training and no extra compute during inference, we were able to produce a better model. (2) While the uniform soup can outperform the best individual model, we only observe this when all individual models achieve high accuracy (e.g., when fine-tuning ALIGN in Figure 4.7); unlike the examples in Figure 4.8, there can be an error barrier between fine-tuned models. We mainly observe this when fine-tuning with high learning rates. However, these high learning rate models also have a lower accuracy, and are therefore excluded by the greedy soup.

Fine-tuning a ViT-G model pre-trained on JFT-3B

To test whether the gains obtained by model soups are additive with other techniques used to obtain state-of-the-art models, we applied our greedy soup technique to 58 ViT-G/14 models fine-tuned on ImageNet. We vary the learning rate, decay schedule, loss function, and minimum crop size in the data augmentation, and optionally apply RandAugment (Cubuk

et al., 2020), mixup (Zhang et al., 2017), or CutMix (Yun et al., 2019). We also train four models with sharpness-aware minimization (SAM) (Foret et al., 2021). For each model training run, we save exponential moving averages (EMA) of the weights (Szegedy et al., 2016) computed with decay factors of 0.999 (low EMA) and 0.9999999 (high EMA). Whereas high EMA generally provides the best single-model accuracy, both greedy soup and greedy ensembling attain higher validation accuracy when applied to parameters with low EMA. We report the highest single model accuracy numbers obtained with either EMA decay value, but perform greedy soup and ensembling with models trained with EMA decay of 0.999. For each combination of training run and EMA decay rate, we evaluate accuracy on our held out validation set every 1000 steps. We use these accuracy values to pick the best checkpoint for ensembling, souping, and subsequent evaluation.

In Table 4.4, we report results on the ImageNet validation set and the five distribution shift datasets studied above as well as two relabeled ImageNet validation sets, ReaL (Beyer et al., 2020) and multilabel (Shankar et al., 2020). Our greedy soup procedure selects 14 of the 58 models fine-tuned as part of our hyperparameter sweep, and this soup performs statistically significantly better than the best individually fine-tuned model selected based on our held out validation set on all datasets except for ObjectNet. Even when we give an unfair advantage to individually fine-tuned models by selecting them based on their performance on each test set (denoted "oracle" in Table 4.4), the greedy soup, which was selected using only in-distribution data, remains superior on most datasets. Only on ReaL and ObjectNet does there exist an individual model that performs statistically significantly better than the soup, and the best model differs between those two datasets. Greedy ensembling performs similarly to the greedy soup in terms of ImageNet top-1 and multilabel accuracy, and slightly better on ReaL, but significantly worse on all distribution shift datasets except for ImageNet-V2. Thus, greedy soup can provide additional gains on top of standard hyperparameter tuning even in the extremely high accuracy regime.

Table 4.5: Performance of model soups on four text classification datasets from the GLUE benchmark (Wang et al., 2018).

Model	Method	MRPC	RTE	CoLA	SST-2
BERT (Devlin et al., 2019b)	Best individual model	88.3	61.0	59.1	92.5
	Greedy soup	88.3 (+0.0)	61.7 (+0.7)	59.1 (+0.0)	93.0 (+0.5)
T5 (Raffel et al., 2020c)	Best individual model	91.8	78.3	58.8	94.6
	Greedy soup	92.4 (+0.6)	79.1 (+0.8)	60.2 (+0.4)	94.7 (+0.1)

Fine-tuning on text classification tasks

To test whether the gains obtained by model soups extend to domains beyond image classification, we conduct preliminary experiments with natural language processing (NLP). While more investigation is warranted to establish the applicability of model soups for NLP, we believe our experiments are a promising initial step. In particular, we fine-tune BERT (Devlin et al., 2019b) and T5 (Raffel et al., 2020c) models on four text classification tasks from the GLUE benchmark (Wang et al., 2018): MRPC (Dolan and Brockett, 2005), RTE (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), CoLA (Warstadt et al., 2019) and SST-2 (Socher et al., 2013), as in (Dodge et al., 2020). We use the standard metric for each dataset: average of accuracy and F_1 score for MRPC, accuracy for RTE, Matthews correlation for CoLA (Matthews, 1975) and accuracy for SST-2.

We fine-tune 32 models for each dataset with a random hyper-parameter search over learning rate, batch size, number of epochs and random seed. Table 4.5 reports the corresponding metric on the validation set for BERT-base uncased (Devlin et al., 2019a) and T5-base (Raffel et al., 2020c). While the improvements are not as pronounced as in image classification, the greedy soup can improve performance over the best individual model in many cases.

4.2.5 Analytically comparing soups to ensembles

The goal of this section is to obtain complementary analytical insight into the effectiveness of model soups. For simplicity, we consider a soup consisting of only two models with

parameters θ_0 and θ_1. For weighting parameter $\alpha \in [0,1]$ we let $\theta_\alpha = (1-\alpha)\theta_0 + \alpha\theta_1$ denote the weight-averaged soup. We would like to understand when the soup error, $\mathrm{err}_\alpha \coloneqq \mathbb{E}_{x,y} 1\{\arg\max_i f_i(x;\theta_\alpha) \neq y\}$, would be lower that the best of both endpoints, $\min\{\mathrm{err}_0, \mathrm{err}_1\}$.

Note that just convexity of err_α in α does not by itself imply superiority of the soup to both endpoints, as the minimum of err_α over α may be obtained at the endpoints even when err_α is convex. To get further leverage on the problem, we compare the soup to the *logit-level ensemble* $f_\alpha^{\mathrm{ens}}(x) = (1-\alpha)f(x;\theta_0) + \alpha f(x;\theta_1)$. The rich literature on ensembles tells us that the expected error of the ensemble, $\mathrm{err}_\alpha^{\mathrm{ens}}$, is often strictly below $\min\{\mathrm{err}_0, \mathrm{err}_1\}$ for neural networks. Therefore, whenever $\mathrm{err}_\alpha \approx \mathrm{err}_\alpha^{\mathrm{ens}}$ we expect the soup to outperform both endpoint models.

To analytically compare the soup and the ensemble, we replace the 0-1 loss with a differentiable surrogate. Specifically, we consider the cross-entropy loss $\ell(f,y) = \log\left(\sum_{y'} e^{f_{y'} - f_y}\right)$. We let $\mathcal{L}_\alpha^{\mathrm{soup}} = \mathbb{E}_{x,y}\ell(\beta f(x;\theta_\alpha), y)$ denote the β-calibrated expected loss of the soup, and similarly define $\mathcal{L}_\alpha^{\mathrm{ens}} = \mathbb{E}_{x,y}\ell(\beta f_\alpha^{\mathrm{ens}}(x), y)$ for the ensemble. We derive the following approximation for the loss difference:

$$\mathcal{L}_\alpha^{\mathrm{soup}} - \mathcal{L}_\alpha^{\mathrm{ens}} \approx \frac{\alpha(1-\alpha)}{2}\left(-\frac{\mathrm{d}^2}{\mathrm{d}\alpha^2}\mathcal{L}_\alpha^{\mathrm{soup}} + \beta^2 \mathbb{E}_x \mathrm{Var}_{Y\sim p_{\mathrm{sftmx}}(\beta f(x;\theta_\alpha))}\left[\Delta f_Y(x)\right]\right),$$

where $[p_{\mathrm{sftmx}}(f)]_i = e^{f_i}/\sum_j e^{f_j}$ is the standard "softmax" distribution and $\Delta f(x) = f(x;\theta_1) - f(x;\theta_0)$ is the difference between the endpoint logits. We obtain our approximation in the regime where the logits are not too far from linear.

The first term in the approximation is negatively proportional to the second derivative of the loss along the trajectory: when the approximation holds, convexity of the loss indeed favors the soup. However, the second term in the approximation does not follow from the "convex basin" intuition. This term always favors the ensemble, but is small in one of two cases: (a) the somewhat trivial case when the endpoint models are similar (so that Δf is small) and (b) when the soup produces confident predictions, implying that $p_{\mathrm{sftmx}}(\beta f(x;\theta_\alpha))$ is close to a point mass and consequently the variance term is small.

To test our approximation, we evaluate it over of set of fine-tuned models with different learning rates, augmentation strategies, random seeds and α values. We set β to calibrate

the soup model, and find that it improves the ability of our approximation to predict the soup/ensemble error difference.

When excluding the high learning rate of 10^{-4} (center and right panels),[3] we see that the approximation is strongly correlated with both the true difference in loss as well as the difference in error, and the approximation and true loss difference generally agree in sign.

4.2.6 Scope and limitations

While this work has so far demonstrated that averaging many fine-tuned models is a useful technique for improving accuracy, this section explores two limitations of the approach. The first is the applicability of model soups, and the second is the failure of model soups to substantially improve calibration.

Applicability. So far our experiments have mainly explored models pre-trained on large, heterogeneous datasets. While the greedy soup still provides improvements on ImageNet, these improvements are less substantial compared to those observed when fine-tuning CLIP and ALIGN.

Calibration. While ensembles improve model calibration (Guo et al., 2017; Roelofs et al., 2020), model soups do not have the same effect. As hyperparameters can also have an effect on calibration, we consider the ensemble and soup of 20 models which are identical other than random seed. We find that, like model ensembling, model soups improve accuracy, but unlike model ensembling, model soups do not improve calibration.

4.2.7 Conclusion

Our results challenge the conventional procedure of selecting the best model on the held-out validation set when fine-tuning. With no extra compute during inference, we are often able to produce a better model by averaging the weights of multiple fine-tuned solutions.

---

[3] Fine-tuned models with learning rate 10^{-4} are far in weight space from the initial model and are often rejected when forming greedy soups. Therefore, we do not expect our approximation to be tight for these learning rates.

4.3 Patching models by interpolating weights

Thanks to advances in large-scale pre-training, recent open-vocabulary models—characterized by their ability to perform any image classification task based on text descriptions of the classes (Pham et al., 2021a)—such as CLIP and BASIC have reached parity with or surpassed important task-specific baselines, even when the open-vocabulary models are not fine-tuned on task-specific data (i.e., in a zero-shot setting) (Radford et al., 2021; Jia et al., 2021; Pham et al., 2021a; Zhai et al., 2022; Alayrac et al., 2022; Yu et al., 2022). For instance, the largest CLIP model from Radford et al. (2021) used in a zero-shot setting matches the ImageNet accuracy of a ResNet-50 trained on 1.2 million ImageNet images (Deng et al., 2009; He et al., 2016).

Nevertheless, current open-vocabulary models still face challenges. The same CLIP model that matches a ResNet-50 on ImageNet has lower MNIST accuracy than simple logistic regression in pixel space (Radford et al., 2021). Moreover, even when zero-shot models achieve good performance, they are usually still worse than models trained or fine-tuned on specific downstream tasks.

To address these issues, several authors have proposed methods for adapting zero-shot models to a task of interest using labeled data (Wortsman et al., 2022b; Zhou et al., 2022; Gao et al., 2021; Zhang et al., 2021; Kumar et al., 2022a; Sung et al., 2022). A common practice is to fine-tune the zero-shot model on the task of interest (Wortsman et al., 2022b; Pham et al., 2021a). However, fine-tuned models can suffer from catastrophic forgetting (McCloskey and Cohen, 1989; Thrun, 1998; French, 1999; Kirkpatrick et al., 2017), performing poorly on tasks where the zero-shot model initially performed well (Andreassen et al., 2021; Wortsman et al., 2022b; Pham et al., 2021a). Additionally, fine-tuning typically produces a task-specific classification head, sacrificing the flexible text-based API that makes open-vocabulary models so appealing. Whereas an open-vocabulary model can perform any classification task in a zero-shot fashion, a fine-tuned model with a task-specific head can only process the specific task that it was fine-tuned on. This specialization can prevent knowledge obtained by fine-tuning on one task from transferring to other related tasks with different classes.

Another approach to adapting zero-shot models would be to add data from the downstream task to the pre-training dataset and train a new open-vocabulary model from scratch. The resulting model could still perform any classification task, and zero-shot performance may improve on related tasks. However, training large image-text models from scratch can require hundreds of thousands of GPU hours (Radford et al., 2021; Pham et al., 2021a; Yu et al., 2022), which makes this approach practically infeasible in most settings.

In this section, we study *patching* open-vocabulary models, where the goal is to increase accuracy on new target tasks while maintaining the flexibility of the model and its accuracy on other tasks.[4] Patching aims to combine the benefits of fine-tuning and re-training from scratch: improved performance on the task of interest, maintaining the flexibility of an open vocabulary, transfer between tasks, and fast adaptation time. Motivated by these goals, we extend existing fine-tuning techniques (Wortsman et al., 2022b) to open-vocabulary settings, where the class space is not fixed. We introduce Patching with Interpolation (**PAINT**), a simple, two-step procedure for patching models: first, fine-tune the model on the patching task without introducing any task-specific parameters; then, linearly interpolate between the weights of the model before and after fine-tuning. Linearly interpolating neural network weights (Nagarajan and Kolter, 2019; Frankle et al., 2020; Neyshabur et al., 2020) has been previously used to improve accuracy on a single task (Izmailov et al., 2018; Wortsman et al., 2022a) or robustness to distribution shift (Wortsman et al., 2022b). Indeed, averaging network weights has been explored in continual learning contexts, although for closed-vocabulary models (Lee et al., 2017).

With **PAINT**, accuracy can improve on new tasks without degrading accuracy on unrelated tasks, as illustrated in Figure 4.12. For instance, applying **PAINT** to a CLIP ViT-L/14 (Radford et al., 2021) independently on nine image classification tasks (Krause et al., 2013; Cimpoi et al., 2014; Helber et al., 2019; Stallkamp et al., 2011; Geiger et al., 2012; LeCun, 1998; Cheng et al., 2017; Xiao et al., 2016; Netzer et al., 2011) improves accuracy by 15 to 60 percentage points compared to the unpatched model, while accuracy on ImageNet (Deng

[4]The term *patching* is borrowed from software development terminology, drawing inspiration from recent work which conceptualizes developing machine learning models like open-source software (Raffel, 2021; Ribeiro and Lundberg, 2022; Matena and Raffel, 2021; Sung et al., 2021).

et al., 2009) decreases by less than one percentage point. We also observe a promising trend: patching becomes more effective with model scale (Section 4.3.3).

Beyond single tasks, we show that models can be patched on multiple tasks (Section 4.3.5). When patching on nine image classification tasks simultaneously, a single CLIP ViT-L/14 model is competitive with using one specialized model for each task—the average accuracy difference is less than 0.5 percentage points.

Moreover, PAINT enables *broad transfer* (Section 4.3.6): accuracy on related tasks can increase, even when the class space changes. For instance, we partition EuroSAT (Helber et al., 2019), a satellite image dataset, into two halves with disjoint labels. Patching a ViT-L/14 model on the first half improves accuracy on the second half by 7.3 percentage points, even though the classes are unseen during patching.

Finally, we investigate PAINT on case studies including typographic attacks (Goh et al., 2021), counting (Johnson et al., 2017), and visual question answering (Antol et al., 2015) (Section 4.3.7). For instance, applying PAINT using synthetic typographic attacks leads to a model that is less susceptible to typographic attacks in the real world, improving its accuracy by 41 percentage points.

In summary:

- Even the best pre-trained models are not perfect. We introduce PAINT, a method designed to improve accuracy on new tasks without harming accuracy elsewhere.
- PAINT incurs no extra computational cost compared to standard fine-tuning, neither during fine-tuning itself nor at inference time.
- PAINT can also be applied with multiple tasks, providing a single model that is competitive with many specialized models.
- Applying PAINT with one task can improve accuracy on a related task, even when they do not share the same classes.
- PAINT improves with model scale, indicating a promising trend for future models.

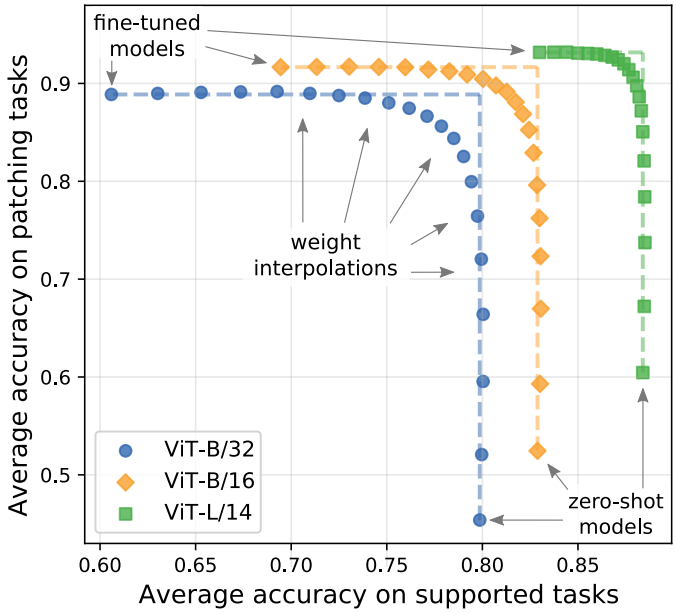

Figure 4.12: **Patching open-vocabulary models by linearly interpolating weights.** We wish to improve accuracy on tasks where a model performs poorly (*patching tasks*), without degrading performance on tasks where accuracy is already adequate (*supported tasks*). When interpolating weights of fine-tuned models and zero-shot (unpatched) models, there are intermediate solutions where accuracy improves on the patching task without reducing accuracy on supported tasks. Results are shown for CLIP models (Radford et al., 2021), averaged over nine patching tasks (Stanford Cars, DTD, EuroSAT, GTSRB, KITTI distance, MNIST, RESISC45, SUN397 and SVHN (Krause et al., 2013; Cimpoi et al., 2014; Helber et al., 2019; Stallkamp et al., 2011; Geiger et al., 2012; LeCun, 1998; Cheng et al., 2017; Coates et al., 2011; Xiao et al., 2016; Netzer et al., 2011)) and five supported tasks (ImageNet, CIFAR-10, CIFAR-100, STL-10 and Food101 (Deng et al., 2009; Krizhevsky et al., 2009; Coates et al., 2011; Bossard et al., 2014)). We apply PAINT separately on each patching task and average results across experiments. The dashed lines illustrate vertical movement from the unpatched models and horizontal movement from the fine-tuned models.

4.3.1 Patching with interpolation (PAINT)

This section details our method for patching models on a single and multiple tasks.

Patching on a single task. Given an open-vocabulary model with weights θ_{zs} and a patching task $\mathcal{D}_{\text{patch}}$, our goal is to produce a new model θ_{patch} which achieves high accuracy on $\mathcal{D}_{\text{patch}}$ without decreasing model performance on tasks where accuracy is already acceptable. We let $\mathcal{D}_{\text{supp}}$ denote a representative supported task where model performance is adequate, and later show that the method is stable under different choices of $\mathcal{D}_{\text{supp}}$ (Section 4.3.4). The two-step procedure we explore for producing θ_{patch} is given below.

Step 1. Fine-tune θ_{zs} on training data from $\mathcal{D}_{\text{patch}}$ to produce a model with weights θ_{ft}.

Step 2. For mixing coefficient $\alpha \in [0, 1]$, linearly interpolate between θ_{zs} and θ_{ft} to produce $\theta_{\text{patch}} = (1 - \alpha) \cdot \theta_{\text{zs}} + \alpha \cdot \theta_{\text{ft}}$. The mixing coefficient is determined via held-out validation sets for $\mathcal{D}_{\text{supp}}$ and $\mathcal{D}_{\text{patch}}$. We refer to the resulting model as θ_{patch}.

In our experiments, we do not introduce any additional task-specific parameters when fine-tuning, as discussed in Section 4.3.2.

Patching on a multiple tasks. In practice, we often want to improve model accuracy on multiple patching tasks $\mathcal{D}^{(1)}_{\text{patch}}, ..., \mathcal{D}^{(k)}_{\text{patch}}$, which can be accomplished with straightforward modifications to the procedure above. We explore three alternatives and examine their relative trade-offs in Section 4.3.5:

- *Joint patching*, where we merge all the patching tasks $\mathcal{D}^{(i)}_{\text{patch}}$ into a single task $\mathcal{D}_{\text{patch}}$ before running the patching procedure;
- *Sequential patching*, where we iteratively repeat the patching procedure above on each new task $\mathcal{D}^{(i)}_{\text{patch}}$ and let $\theta_{\text{zs}} \leftarrow \theta_{\text{patch}}$ after each completed iteration;
- *Parallel patching*, where we apply the first step on each task in parallel to produce fine-tuned models with weights $\theta^{(1)}_{\text{ft}}, ..., \theta^{(k)}_{\text{ft}}$. Then, we search for mixing coefficients α_i to produce $\theta_{\text{patch}} = (1 - \sum_{i=1}^{k} \alpha_i) \cdot \theta_{\text{zs}} + \sum_{i=1}^{k} \alpha_i \cdot \theta^{(i)}_{\text{ft}}$.

For joint and parallel patching we assume access to held-out validation sets for all tasks, while in sequential patching we only assume access to held-out validation sets from the tasks

seen so far. Unless mentioned otherwise, we pick the mixing coefficient α that optimizes average accuracy on the held-out validation sets from the supported and patching tasks.

4.3.2 Experimental setup

Tasks. We consider a diverse set of image classification tasks from Radford et al. (2021). In most experiments, we use ImageNet (Deng et al., 2009) as a representative supported task, although we explore other supported tasks in Section 4.3.4. We categorize tasks into patching tasks or supported tasks based on the accuracy difference between the zero-shot model and a model specialized to the task. A large accuracy difference indicates that the task is a relevant target for patching because the zero-shot model is still far from optimal. Specifically, we consider a subset tasks from Radford et al. (2021), categorizing tasks where the linear probes outperform the zero-shot model by over 10 percentage points as patching tasks: Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2011), KITTI (Geiger et al., 2012), MNIST (LeCun, 1998), RESISC45 (Cheng et al., 2017), SUN397 (Xiao et al., 2016), and SVHN (Netzer et al., 2011). We use the remaining tasks as supported tasks: CIFAR10 (Krizhevsky et al., 2009), CIFAR100 (Krizhevsky et al., 2009), Food101 (Bossard et al., 2014), ImageNet (Deng et al., 2009), and STL10 (Coates et al., 2011). We investigate additional patching tasks as case studies in Section 4.3.7.

Models. We primarily use CLIP (Radford et al., 2021) pre-trained vision transformer (ViT) models (Dosovitskiy et al., 2021). Unless otherwise mentioned our experiments are with the ViT-L/14 model, while Section 4.3.4 studies ResNets (He et al., 2016).

Fine-tuning on patching tasks. Unless otherwise mentioned, we fine-tune with a batch size of 128 for 2000 iterations using learning rate 1e-5 with 200 warm-up steps with a cosine annealing learning rate schedule and the AdamW optimizer (Loshchilov and Hutter, 2019; Paszke et al., 2019) (weight decay 0.1). When fine-tuning, we use the frozen final classification layer output by CLIP's text tower so that we do not introduce additional learnable parameters. This design decision keeps the model open-vocabulary and does not

harm accuracy.

Evaluation. We use accuracy as the evaluation metric unless otherwise stated. We refer to the average of the mean accuracy on the patching tasks and the mean accuracy on the supported tasks as *combined accuracy*.[5]

4.3.3 Patching models on a single new task

As shown in Figure 4.12, when patching a model on a single task, we interpolate the weights of the zero-shot and fine-tuned model, producing a model that achieves high accuracy on both the patching task and the supported task. On the nine tasks, PAINT improves the accuracy of ViT-L/14 by 15 to 60 percentage points, while accuracy on ImageNet decreases by less than one percentage point. PAINT also allows practitioners to control the accuracy trade-off on the patching and supported tasks without re-training a new model, by varying the mixing coefficient α.

The effect of scale

We consistently observe that PAINT is more effective for larger models. Our findings are aligned with those of Ramasesh et al. (2021), who observed that larger models are less susceptible to catastrophic forgetting. This section formalizes and provides insights for these observations.

Measuring the effectiveness of patching. We measure the effectiveness of patching via the accuracy difference between the single patched model and two specialized models with the same architecture and initialization. For both the supported task and patching task, we take specialized models that maximize performance on the task, considering the set of all interpolations between the zero-shot and fine-tuned models. We refer to this measure

[5] In other words, $(\mathbb{E}_{\mathcal{D}_{\text{supp}}}[\mathsf{Acc}(\theta, \mathcal{D}_{\text{supp}})] + \mathbb{E}_{\mathcal{D}_{\text{patch}}}[\mathsf{Acc}(\theta, \mathcal{D}_{\text{patch}})])/2$, where $\mathsf{Acc}(\theta, \mathcal{D}_{\text{supp}})$ and $\mathsf{Acc}(\theta, \mathcal{D}_{\text{patch}})$ are accuracies on supported tasks and patching tasks, respectively.

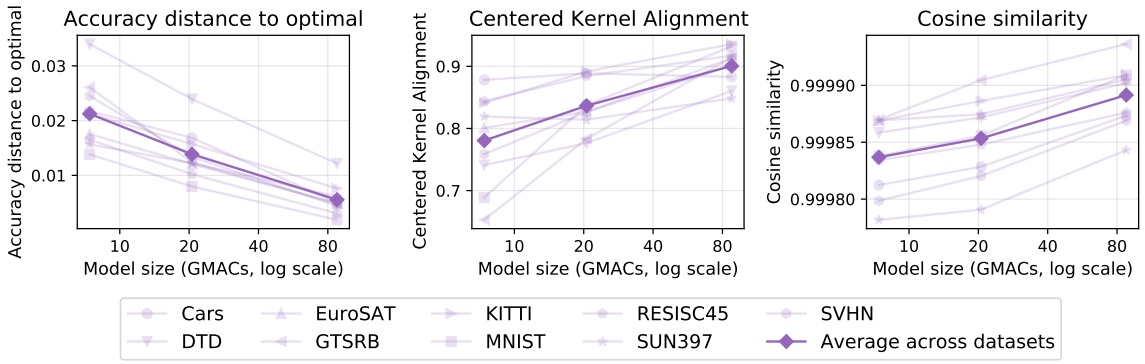

Figure 4.13: **Larger models are easier to patch** (left). For larger models, the unpatched and fine-tuned model are more similar with respect to their representations (center) and weights (right). Model scale is measured in Giga Multiply-Accumulate operations (GMACs).

as *accuracy distance to optimal*. Formally, accuracy distance to optimal is given by

$$\frac{1}{2}\left[\max_{\alpha}\mathsf{Acc}(\theta_\alpha,\mathcal{D}_{\text{supp}})+\max_{\alpha}\mathsf{Acc}(\theta_\alpha,\mathcal{D}_{\text{patch}})\right]-\frac{1}{2}\max_{\alpha}\left[\mathsf{Acc}(\theta_\alpha,\mathcal{D}_{\text{supp}})+\mathsf{Acc}(\theta_\alpha,\mathcal{D}_{\text{patch}})\right], \tag{4.3}$$

where $\mathrm{Acc}(\theta,\mathcal{D})$ represents the accuracy of model θ on task $\mathcal{D}$. In Figure 4.13 (left), we show that accuracy distance to optimal decreases with scale, indicating that patching becomes more effective for larger models.

Model similarity. Fine-tuning modifies overparameterized models less (Chizat et al., 2019), which provides insights on why larger models are easier to patch: less movement is required to fit new data. We demonstrate this by evaluating representational similarity using Centered Kernel Alignment (CKA) (Kornblith et al., 2019a). As shown in Figure 4.13 (center), the representations of the unpatched and fine-tuned models become more similar as models grow larger, indicated by larger CKA values. Moreover, Figure 4.13 (right) shows that the cosine similarity between the weights of the unpatched and fine-tuned models, $\cos(\theta_{\text{zs}},\theta_{\text{ft}}) = \langle\theta_{\text{zs}},\theta_{\text{ft}}\rangle/(||\theta_{\text{zs}}||\,||\theta_{\text{ft}}||)$, increases with scale.

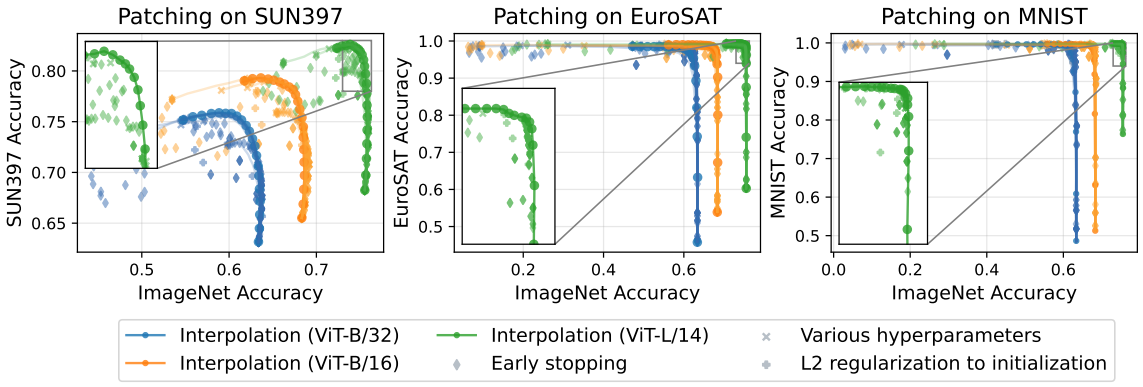

Figure 4.14: **The frontier of accuracy trade-offs can be recovered by linearly interpolating weights.** Interpolating the unpatched and fine-tuned models recovers the accuracy trade-off of early stopping, regularization towards the initialization, and changes in hyperparameters.

4.3.4 Baselines and ablations

Baselines. There are many alternatives which enable a trade-off between accuracy on the supported and patching tasks. These methods include early stopping during fine-tuning, applying a regularization term which penalizes movement from initialization, or training with different hyperparameters including a smaller learning rate. Unlike interpolation, these methods do not enable navigating the accuracy trade-off without fine-tuning the model again many times. Moreover, Figure 4.14 demonstrates that the accuracy trade-off frontier for early stopping, regularization, or varying hyperparameters can be recovered by interpolating weights with different mixing coefficients.

Additional supported tasks. In Figure 4.12, we use ImageNet as a representative supported task. This section demonstrates that PAINT is stable under different choices of the supported task. Instead of ImageNet, we use CIFAR10, CIFAR100, Food101 and STL10. Figure 4.15 displays representative results, where performance is averaged over the nine patching tasks. We observe consistent results across supported tasks, and that the optimal mixing coefficients are stable across different choices of supported tasks (Figure 4.15, right).

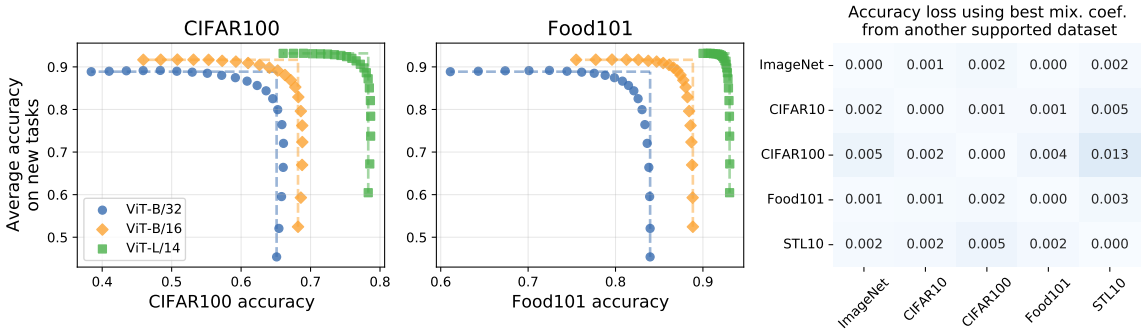

Figure 4.15: **Results are consistent across supported tasks.** For multiple supported tasks, we observe similar accuracy improvements on patching tasks, without substantially decreasing supported task accuracy. Moreover, choosing the mixing coefficients using a different supported task does not substantially decrease combined accuracy on patching and supported tasks (right).

4.3.5 Patching models on multiple tasks

This section details experimental results for patching on multiple datasets. Recall from Section 4.2.1 that there are various strategies for extending **PAINT** to multiple datasets, which we briefly revisit. For *joint* patching we merge all the datasets into a single fine-tuning task and apply our patching procedure as before. For *sequential* patching we iteratively perform our procedure once per task, using the patched model at each step as the initialization for the next step.[6] We also explore *parallel* patching, for which we have an unpatched model θ_{zs} and independently fine-tune on each of the tasks in parallel. We then search for mixing coefficients to combine the resulting models. For tasks $1, ..., k$, let $\theta_{\text{ft}}^{(1)}, ..., \theta_{\text{ft}}^{(k)}$ denote the fine-tuned models for each task. Since it is impractical to exhaustively search over each α_i, we instead search over a one-dimensional scalar $\alpha \in [0, 1]$, which interpolates between θ_{zs} and the average of all fine-tuned solutions $\frac{1}{k}\sum_{i=1}^{k}\theta_{\text{ft}}^{(i)}$.[7]

These methods have various trade-offs and may be applicable for different scenarios.

[6]The results are averaged over three random seeds that control the order in which tasks are seen.

[7]We also explored adaptive black-box optimization algorithms to choose the mixing coefficients α_i (Rapin and Teytaud, 2018), but observed little improvement (0.3 to 0.4 percentage points on average).

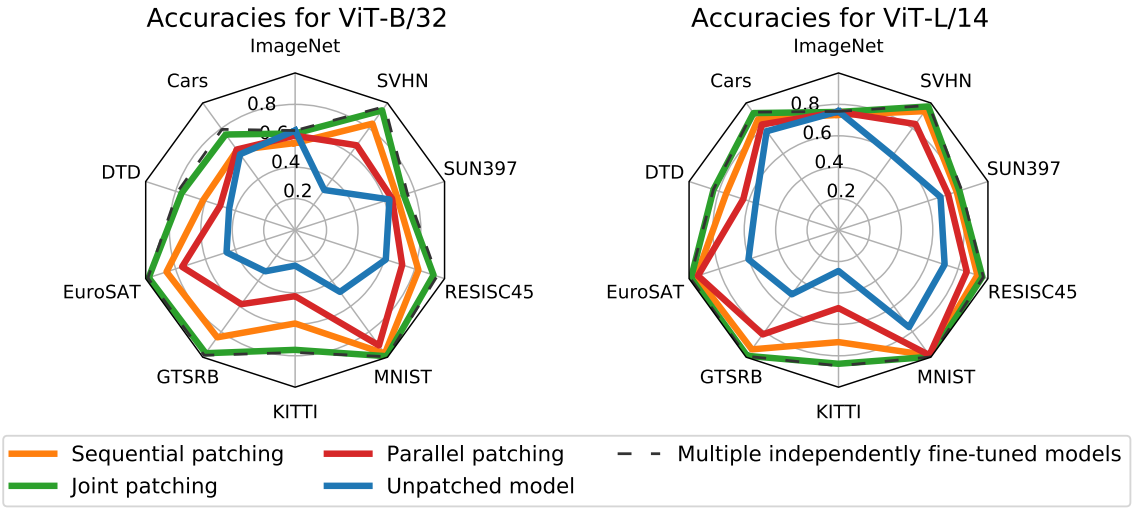

Figure 4.16: **Contrasting various strategies for patching on multiple tasks.** On all experiments, ImageNet is used as the supported task while the other nine datasets are used for patching. When data from all patching tasks is available, joint patching yields a single model that is competitive with using ten different specialized models. Weight interpolations greatly mitigate catastrophic forgetting on the sequential case, but do not completely eradicate it. Finally, parallel patching underperforms other patching strategies, but still provides improvements over the unpatched model.

Joint patching is only possible when data from all tasks you wish to patch is available. On the other hand, sequential patching is appropriate when the tasks are observed one after another. Finally, parallel patching can leverage distributed hardware.

Figure 4.16 displays experimental results when patching on all nine tasks from Section 4.3.3. We observe that joint patching is the best-performing method on average. This is perhaps unsurprising since joint patching has simultaneous access to all patching datasets, unlike other patching strategies. Nevertheless, it is still interesting that for ViT-L/14, joint patching yields a *single* model with only 0.5 percentage points worse combined accuracy than using *multiple* specialized models.[8] Joint patching also achieves a 15.8 percentage points improvement over the unpatched model. Moreover, patching a ViT-B/32 model

[8]Recall from Section 4.2.2 that combined accuracy weight patching and supported tasks equally.

	Cars	DTD	EuroSAT	GTSRB	KITTI	MNIST	RESISC45	SUN397	SVHN
Unpatched accuracy	86.2	64.9	79.9	51.7	43.4	82.6	73.4	76.9	72.8
Patched accuracy	87.0 **(+0.8)**	66.1 **(+1.2)**	87.2 **(+7.3)**	71.1 **(+19.4)**	60.4 **(+17.0)**	91.3 **(+8.7)**	74.2 **(+0.8)**	79.3 **(+2.4)**	88.9 **(+16.1)**

Table 4.6: **PAINT can generalize to unseen classes.** We randomly partition each dataset into tasks A and B with disjoint class spaces of roughly equal size. This table reports how patching on task A affects accuracy on task B for the ViT-L/14 model. In all cases, accuracy on task B improves when patching on task A even though the classes are *unseen* during patching.

with the joint strategy achieves a combined accuracy 6.1 percentage points higher than a ViT-L/14 unpatched model, which requires 12x more GMACs.

The accuracy of sequential patching approaches that of joint patching, especially for larger models. Note that, unlike in joint patching, forgetting can compound since the patching procedure is applied multiple times in sequence. In sequential patching, weight interpolations do not completely eradicate forgetting, but greatly mitigate it. This is most noticeable for smaller models: sequentially fine-tuning a ViT-B/32 without interpolation *reduces* the combined accuracy by 4.6 percentage points compared to the unpatched model. This is compared to a combined accuracy *increase* of 11 percentage points when using sequential patching.

Finally, parallel patching underperforms other patching strategies. Like sequential patching, parallel patching is in the challenging setting where data from all patching tasks is not available simultaneously. Moreover, unlike in joint or sequential patching, no model is optimized on data from all patching tasks. Using a black box optimization algorithm for finding the mixing coefficients did not yield large improvements over using the same mixing coefficient for all models. However, it is possible that more sophisticated search methods could yield better results.

Task A	MNIST	SVHN	EuroSAT	RESISC45	MNIST	FashionMNIST	GTSRB	MTSD
Task B	SVHN	MNIST	RESISC45	EuroSAT	FashionMNIST	MNIST	MTSD	GTSRB
Unpatched accuracy	58.6	76.4	71.0	60.2	67.7	76.4	19.3	50.6
Patched accuracy	68.9	93.2	69.7	70.4	70.8	77.5	30.8	69.8
	(+10.3)	**(+16.8)**	**(-1.3)**	**(+10.2)**	**(+3.1)**	**(+1.1)**	**(+11.5)**	**(+19.2)**

Table 4.7: **Patching on task A can improve accuracy on a related task B.** For a pair of tasks A and B, we report accuracy of the ViT-L/14 on task B, after patching on task A, finding improvements on seven out of eight cases.

4.3.6 Broad transfer

An alternative to our patching approach is to introduce parameters which are specific to each new task. By contrast, PAINT always maintains a single model. This section describes an additional advantage of the single model approach: patching the model on task A can improve accuracy on task B, even when task A and B do not share the same classes. We refer to this phenomenon as *broad transfer*. Note that we are able to study this phenomenon because the single patched model remains open-vocabulary throughout the patching procedure. This is a key advantage of PAINT compared to maintaining a collection of task-specific models.

We now describe two experiments to measure the effects on a task B when patching the model on a task A. First, we explore broad transfer by randomly partitioning datasets into disjoint sets with no class overlap. For a dataset $\mathcal{D}$ we partition the class space $\mathcal{Y}$ into two disjoint sets of roughly equal size $\mathcal{Y}_A$ and $\mathcal{Y}_B$. We build task A with the examples $(x, y) \in \mathcal{D}$ where y belongs to $\mathcal{Y}_A$, and task B with examples (x, y) where y belongs to $\mathcal{Y}_B$. Table 4.6 shows how patching a model on task A affects the accuracy on task B for nine datasets $\mathcal{D}$. The accuracy improvements on task B range from 0.8 to 19.4 percentage points, even though the classes from task B are not seen during patching.

To further understand transfer, we consider additional task pairs A and B, which are now different datasets. While some pairs A, B share classes, there are still instances of

broad transfer. Concretely, Table 4.7 examines i) MNIST and SVHN, two digit recognition tasks with shared classes; ii) EuroSAT and RESISC45, two satellite imagery recognition tasks where there are unshared classes but some overlap; iii) GTSRB and MTSD (Ertler et al., 2020), two traffic sign recognition datasets where there are unshared classes but some overlap; and iv) MNIST and FashionMNIST (Xiao et al., 2017), which do not share any classes but appear visually similar. In seven out of eight experiments, patching on task A improves accuracy by 1.1 to 19.2 percentage points on task B. The exception is when A is EuroSAT and B is RESISC45, where accuracy decreases by 1.3 percentage points.

In all experiments, when patching on task A we choose the mixing coefficient α by optimizing the held-out validation accuracy on task A and a supported task (in this experiment we use ImageNet). While it is possible for a method that introduces new parameters for each task to exhibit broad transfer to new data, this also requires knowing which parameters to apply for the new data. This is not necessary in the single model approach.

4.3.7 Case studies

We further examine the performance of **PAINT** in three additional settings, which highlight weaknesses of the zero-shot CLIP model and showcase broad transfer (Section 4.3.6).

Typographic attacks. Goh et al. (2021) find that CLIP models are susceptible to *typographic attacks*, where text superimposed on an image leads to misclassification. For example, in Figure 4.17 *(a)*, the text on the pink note saying "dog" leads a CLIP to misclassify the image of a cat as a dog. To fix this vulnerability, we procedurally generate typographic attack data by adding text with incorrect class names to SUN397 (Xiao et al., 2016), as seen in Figure 4.17 *(b)*. We then collect a test set of 110 real world images by placing notes on objects and taking photos.[9] After applying **PAINT** using the synthetic data, we evaluate on the real-world images (Figure 4.17 *(c)*) and synthetic test set (Figure 4.17 *(d)*). We observe that while larger models are more susceptible to typographic attacks, they are also more amenable to patching. Furthermore, we see an example of broad transfer between the synthetic and real-world data: when patching ViT-L/14 on synthetic data, its accuracy

[9]Data available at `https://github.com/mlfoundations/patching`.

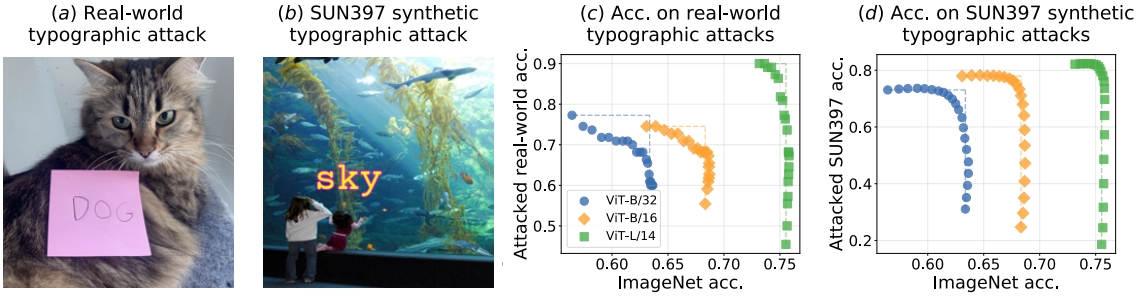

Figure 4.17: **Guarding against real-world typographic attacks by patching on synthetic data.** *(a)* A sample from our real-world typographic attacks test set. A CLIP ViT-L/14 is "tricked" into classifying this image as a dog instead of a cat. *(b)* Sample of synthetic typographic attack data. *(c)* Performance on real-world data with unseen classes after patching on *only* synthetic typographic attacks (curves produced by interpolating between the unpatched and fine-tuned model). *(d)* Analogous curves for the test set of the synthetic data used for patching.

on real-world typographic attacks improves 41 percentage points even though the real-world classes are unseen. The cost is a reduction of less than 1 percentage point on ImageNet.

Counting. Radford et al. (2021) find that CLIP models struggle to count the number of objects in CLEVR (Johnson et al., 2017). Here, the task is to choose an integer between 3 and 10 for each image, corresponding to the number of visible objects. While a straightforward way to patch such a task is to fine-tune on it directly, we investigate if applying PAINT using a subset of the classes allows the patched model to generalize to other numbers. Specifically, we patch on images with 4, 5, 6, 8, or 9 objects. To evaluate broad transfer, we test on images with 3, 7, and 10 objects (7 for understanding interpolation and 3 and 10 for extrapolation). We find that PAINT improves accuracy from 59% to over 99% on unseen classes with less than half a percentage point decrease in ImageNet accuracy.

Visual question answering. As shown by Shen et al. (2022), zero-shot CLIP models perform poorly on visual question answering (Antol et al., 2015). Using CLIP for VQA typ-

ically involves additional parameters—for instance, Shen et al. (2022) trains a transformer (Vaswani et al., 2017) on CLIP features. In contrast, our procedure for patching CLIP on VQA does not introduce new parameters. Following Shen et al. (2022), we contrast images with a series of text prompts, where each prompt corresponds to an option in multiple-choice VQA, formed by both the question and a candidate answer using the following template: "Question: [question text] Answer: [answer text]". We evaluate on multiple-choice VQA v1 (Antol et al., 2015), where each question is associated with 18 candidate answers. Our results show that patching is effective for visual question answering: PAINT improves the accuracy of a ViT-L/14 model by 18 percentage points, while accuracy drops by less than one percentage point on ImageNet.

4.3.8 Limitations and conclusion

Limitations. When applying PAINT, accuracy on supported tasks can still decrease, especially for smaller models. This limitation is perhaps best reflected in the case of sequential patching: patched models underperform using multiple specialized models when many tasks are added sequentially. Using larger models and weight interpolations can alleviate this issue, but do not completely resolve it. Finally, better understanding on which datasets patching is more effective is an exciting direction for future research.

Conclusion. In this section, we explore several techniques for patching open-vocabulary models with the goal of improving accuracy on new tasks without decreasing accuracy elsewhere. PAINT is effective in several scenarios, ranging from classifying digits to defending against typographic attacks. PAINT becomes more effective with scale, and can be applied on multiple tasks sequentially or simultaneously. Our findings demonstrate that in many circumstances it is possible to expand the set of tasks on which models achieve high accuracy, without introducing new parameters, without re-training them from scratch, and without catastrophic forgetting.

4.4 Editing Models with Task Arithmetic

In this section, we present a new paradigm for editing neural networks based on *task vectors*, which encode the information necessary to do well on a given task. Inspired by recent work on weight interpolation (Frankle et al., 2020; Wortsman et al., 2022b; Matena and Raffel, 2021; Wortsman et al., 2022a; Ilharco et al., 2022; Li et al., 2022; Ainsworth et al., 2022; Don-Yehiya et al., 2022), we obtain such vectors by taking the weights of a model fine-tuned on a task and subtracting the corresponding pre-trained weights (Figure 4.18a).

We show that we can edit a variety of models with *task arithmetic*—performing simple arithmetic operations on task vectors (Figure 4.18b-d). For example, *negating* a vector can be used to remove undesirable behaviors or unlearn tasks, while *adding* task vectors leads to better multi-task models, or even improves performance on a single task. Finally, when tasks form an *analogy* relationship, task vectors can be combined to improve performance on tasks where data is scarce.

Forgetting via negation. Users can negate task vectors to mitigate undesirable behaviors (e.g., toxic generations), or even to forget specific tasks altogether, like OCR. In Section 4.4.2, we negate a task vector from a language model fine-tuned on toxic data (Radford et al., 2019; Borkan et al., 2019), reducing the proportion of generations classified as toxic, with little change in fluency. We also negate task vectors for image classification tasks, resulting in substantially lower accuracy on the task we wish to forget with little loss on ImageNet accuracy (Deng et al., 2009).

Learning via addition. Adding task vectors results in better multi-task models, or improved performance on a single task. In Section 4.4.3, we add task vectors from various image models (CLIP, Radford et al. (2021)) and compare the performance of the resulting model with using multiple specialized fine-tuned models. We find that the single resulting model can be competitive with using multiple specialized models. Adding two task vectors maintains 98.9% of the accuracy, and the average performance on the entire set of tasks increases as more task vectors are added. Moreover, adding a task vector from a different task can *improve* performance on a target task using text models (T5, Raffel et al. (2020a)).

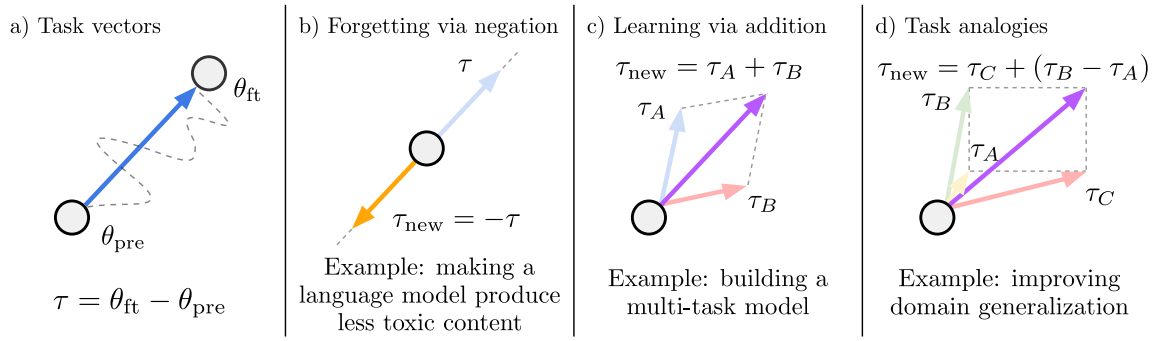

Figure 4.18: An illustration of task vectors and the arithmetic operations we study for editing models. (a) A task vector is obtained by subtracting the weights of a pre-trained model from the weights of the same model after fine-tuning (Section 4.4.1). (b) Negating a task vector degrades performance on the task, without substantial changes in control tasks (Section 4.4.2). (c) Adding task vectors together improves the performance of the pre-trained model on the tasks under consideration (Section 4.4.3). (d) When tasks form an analogy relationship such as supervised and unsupervised learning on two different data sources, it is possible to improve performance on a supervised target task using only vectors from the remaining three combinations of objectives and datasets (Section 4.4.4).

Task analogies. When we can form task analogies of the form "A is to B as C is to D", combining task vectors from the first three tasks improves performance on the fourth, even when little or no training data is available. In Section 4.4.4, we show that we can improve domain generalization to a new target task without using labeled data from that task. More specifically, accuracy on a sentiment analysis task improves by combining a task vector from a second sentiment analysis dataset and task vectors produced using unlabeled data from both domains. We also use analogies between classifying pictures and sketches of objects to improve accuracy on subgroups where little or no data is available.

Overall, editing models with task arithmetic is simple, fast and effective. There is no extra cost at inference time in terms of memory or compute, since we only do element-wise operations on model weights. Moreover, vector operations are cheap, allowing users to experiment quickly with multiple task vectors. With task arithmetic, practitioners can

reuse or transfer knowledge from models they create, or from the multitude of publicly available models all without requiring access to data or additional training.[10]

4.4.1 Task Vectors

For our purposes, a task is instantiated by a dataset and a loss function used for fine-tuning. Let $\theta_{\text{pre}} \in \mathbb{R}^d$ be the weights of a pre-trained model, and $\theta_{\text{ft}}^t \in \mathbb{R}^d$ the corresponding weights after fine-tuning on task t. The task vector $\tau_t \in \mathbb{R}^d$ is given by the element-wise difference between θ_{ft}^t and θ_{pre}, i.e., $\tau_t = \theta_{\text{ft}}^t - \theta_{\text{pre}}$. When the task is clear from context, we omit the identifier t, referring to the task vector simply as τ.

Task vectors can be applied to any model parameters θ from the same architecture, via element-wise addition, with an optional scaling term λ, such that the resulting model has weights $\theta_{\text{new}} = \theta + \lambda\tau$. In our experiments, the scaling term is determined using held-out validation sets. Note that adding a single task vector to a pre-trained model with $\lambda = 1$ results in the model fine-tuned on that task.

Following Ilharco et al. (2022), we focus on open-ended models, where it is possible to fine-tune on a downstream task without introducing new parameters (e.g., open-vocabulary image classifiers (Radford et al., 2021; Jia et al., 2021; Pham et al., 2021a; Alayrac et al., 2022) and text-to-text models (Raffel et al., 2020a; Radford et al., 2019; Brown et al., 2020b; Hoffmann et al., 2022b)). In cases where fine-tuning introduces new parameters (e.g., a new classification head), we could follow Matena and Raffel (2021) and merge only the shared weights, but this exploration is left for future work.

Editing models with task arithmetic. We focus on three arithmetic expressions over task vectors, as illustrated in Figure 4.18: negating a task vector, adding task vectors together, and combining task vectors to form analogies. All operations are applied element-wise to the weight vectors.

When *negating* a task vector τ, applying the resulting vector $\tau_{\text{new}} = -\tau$ corresponds to extrapolating between the fine-tuned model and the pre-trained model. The resulting model

[10] Code available at https://github.com/mlfoundations/task_vectors.

is worse at the target task, with little change in performance on control tasks (Section 4.4.2). *Adding* two or more task vectors τ_i yields $\tau_{\text{new}} = \sum_i \tau_i$, and results in a multi-task model proficient in all tasks, sometimes even with gains over models fine-tuned on individual tasks (Section 4.4.3). Finally, when tasks A, B, C and D form an analogy in the form "A is to B as C is to D", the task vector $\tau_{\text{new}} = \tau_C + (\tau_B - \tau_A)$ improves performance on task D, even if there is little or no data for that task (Section 4.4.4).

For all operations, the model weights obtained by applying τ_{new} are given by $\theta_{\text{new}} = \theta + \lambda\tau_{\text{new}}$, where the scaling term λ is determined using held-out validation sets.

4.4.2 Forgetting via Negation

In this section, we show that negating a task vector is an effective way to reduce its performance on a target task, without substantially hurting performance elsewhere. Forgetting or "unlearning" can help mitigate undesired biases learned when pre-training; forgetting tasks altogether may be desirable to comply with regulations or for ethical reasons like preventing an image classifier to recognize faces, or to "read" personal information via OCR.

These interventions should not have a substantial effect on how models behave when processing data outside the scope of the edit (Mitchell et al., 2021; Ilharco et al., 2022). Accordingly, we measure accuracy on *control tasks*, in addition to evaluating on the target tasks from which the task vector originated. Our experiments showcase the effectiveness of negating task vectors for editing image classification and text generation models.

Image classification

For image classification, we use CLIP models (Radford et al., 2021) and task vectors from eight tasks studied by Ilharco et al. (2022); Radford et al. (2021), ranging from satellite imagery recognition to classifying traffic signs: Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2011), MNIST (LeCun, 1998), RESISC45 (Cheng et al., 2017), SUN397 (Xiao et al., 2016), and SVHN (Netzer et al., 2011). For the control task, we use ImageNet (Deng et al., 2009). We generate task vectors by fine-tuning on each of the target tasks.

Table 4.8: **Forgetting image classification tasks via negation**. Results are shown for CLIP models, reporting average accuracy (%) on the eight target tasks we wish to forget (Cars, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397 and SVHN), and the control task (ImageNet). Negating task vectors reduce the accuracy of a pre-trained ViT-L/14 by 45.8 percentage points on the target tasks, with little loss on the control task.

Method	ViT-B/32		ViT-B/16		ViT-L/14	
	Target (↓)	Control (↑)	Target (↓)	Control (↑)	Target (↓)	Control (↑)
Pre-trained	48.3	63.4	55.2	68.3	64.8	75.5
Fine-tuned	90.2	48.2	92.5	58.3	94.0	72.6
Gradient ascent	2.73	0.25	1.93	0.68	3.93	16.3
Random vector	45.7	61.5	53.1	66.0	60.9	72.9
Negative task vector	24.0	60.9	21.3	65.4	19.0	72.9

We compare against two additional baselines, fine-tuning by moving in the direction of increasing loss (i.e., with gradient ascent), as in Golatkar et al. (2020); Tarun et al. (2021), and against using a random vector where each layer has the same magnitude as the corresponding layer of task vector.

As shown in Table 4.8, negating the task vectors is the most effective editing strategy for decreasing accuracy on the target task with little impact on the control task. For example, negative task vectors decrease the average target accuracy of ViT-L/14 by 45.8 percentage points with little change in accuracy on the control task. In contrast, using a random vector does not have much impact on target accuracy, while fine-tuning with gradient ascent severely deteriorates performance on control tasks.

Text generation

We study whether we can mitigate a particular model behavior by negating a task vector *trained to do that behavior*. In particular, we aim to reduce the amount of toxic generations produced by GPT-2 models of various sizes (Radford et al., 2019). We generate task vectors by fine-tuning on data from Civil Comments (Borkan et al., 2019) where the toxicity score is

Table 4.9: **Making language models less toxic with negative task vectors.** Results are shown for the GPT-2 Large model. Negative task vectors decrease the amount of toxic generations by 6×, while resulting in a model with comparable perplexity on a control task (WikiText-103).

Method	% toxic generations (↓)	Avg. toxicity score (↓)	WikiText-103 perplexity (↓)
Pre-trained	4.8	0.06	16.4
Fine-tuned	57	0.56	16.6
Gradient ascent	0.0	0.45	$>10^{10}$
Fine-tuned on non-toxic	1.8	0.03	17.2
Random vector	4.8	0.06	16.4
Negative task vector	0.8	0.01	16.9

higher than 0.8, and then negating such task vectors. As in Section 4.4.2, we also compare against baselines that use gradient ascent when fine-tuning (Golatkar et al., 2020; Tarun et al., 2021), and using a random task vector of the same magnitude. Additionally, we compare against fine-tuning on non-toxic samples from Civil Comments (toxicity scores smaller than 0.2), similar to Liu et al. (2021). We measure the toxicity of one thousand model generations with Detoxify (Hanu and Unitary team, 2020). For the control task, we measure the perplexity of the language models on WikiText-103 (Merity et al., 2016).

As shown in Table 4.9, editing with negative task vectors is effective, reducing the amount of generations classified as toxic from 4.8% to 0.8%, while maintaining perplexity on the control task within 0.5 points of the pre-trained model. In contrast, fine-tuning with gradient ascent lowers toxic generations by degrading performance on the control task to an unacceptable level, while fine-tuning on non-toxic data is worse than task vectors both in reducing task generations and on the control task. As an experimental control, adding a random vector has little impact either on toxic generations or perplexity on WikiText-103.

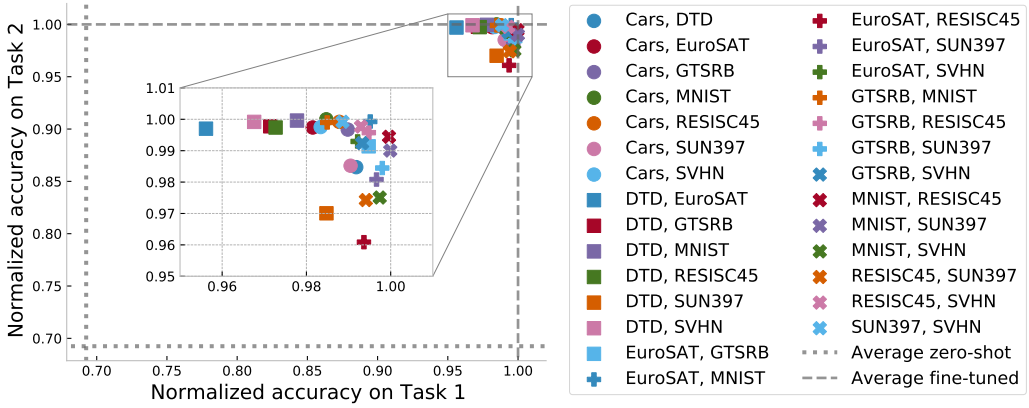

Figure 4.19: **Adding pairs of task vectors** from image classification tasks. Adding task vectors from two tasks improves accuracy on both, resulting in a single model that is competitive with using two specialized fine-tuned models.

4.4.3 Learning via Addition

We now turn our attention to *adding* task vectors, either to build multi-task models that are proficient on multiple tasks simultaneously, or to improve single-task performance. This operation allows us to reuse and transfer knowledge either from in-house models, or from the multitude of publicly available fine-tuned models, without additional training or access to training data. We explore addition on various image classification and natural language processing tasks.

Image classification

We start with the same eight models used in Section 4.4.2, fine-tuned on a diverse set of image classification tasks (Cars, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397 and SVHN). In Figure 4.19, we show the accuracy obtained by adding all pairs of task vectors from these tasks. To account for the difference in difficulty of the tasks, we normalize accuracy on each task by the accuracy of the model fine-tuned on that task. After normalizing, the performance of fine-tuned models on their respective tasks is one, and so the average performance of using multiple specialized models is also one. As shown in Figure 4.19,

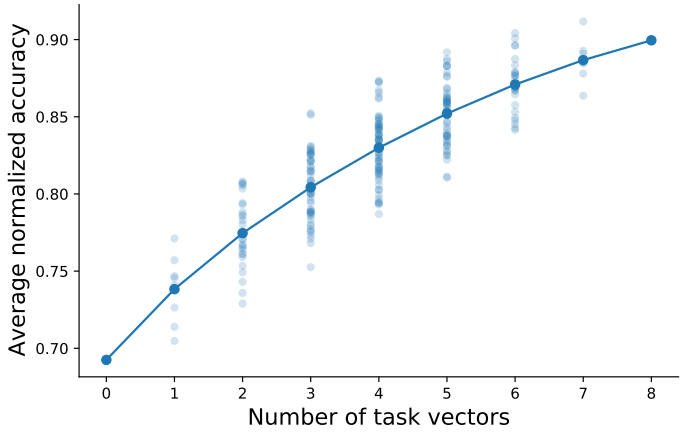

Figure 4.20: **Adding task vectors builds multi-task models** for image classification tasks. Accuracy is averaged over all downstream tasks. When more task vectors are available, better multi-task vectors can be built. Each point represents an experiment with a subset of the eight tasks we study, and the solid line connects the average performance for each subset size. Recall that the average normalized accuracy of using multiple fine-tuned models is always one.

adding pairs of task vectors leads to a single model that outperforms the zero-shot model by a large margin, and is competitive with using two specialized models (98.9% normalized accuracy on average).

Beyond pairs of tasks, we explore adding task vectors for *all* possible subsets of the tasks (2^8 in total). In Figure 4.20, we show how the normalized accuracy of the resulting models, averaged over all the eight tasks. As the number of available task vectors increases, better multi-task models can be produced. When all task vectors are available, the best model produced by adding task vectors reaches an average performance of 91.2%, despite compressing several models into one.

Natural language processing

In addition to building multi-task models, we explore whether adding task vectors is a useful way of improving performance on a single target task. Towards this goal, we first

Table 4.10: **Improving performance on target tasks with external task vectors.** For four text classification tasks from the GLUE benchmark, adding task vectors downloaded from the Hugging Face Hub can improve accuracy of fine-tuned T5 models.

Method	MRPC	RTE	CoLA	SST-2	Average
Zero-shot	74.8	52.7	8.29	92.7	57.1
Fine-tuned	88.5	77.3	52.3	94.5	78.1
Fine-tuned + task vectors	89.3 (+0.8)	77.5 (+0.2)	53.0 (+0.7)	94.7 (+0.2)	78.6 (+0.5)

fine-tune T5-base models on four tasks from the GLUE benchmark (Wang et al., 2018), as in Wortsman et al. (2022a). Then, we search for compatible checkpoints on Hugging Face Hub, finding 427 candidates in total. We try adding each of the corresponding task vectors to our fine-tuned models, choosing the best checkpoint and scaling coefficient based on held-out validation data. As shown in Table 4.10, adding task vectors can *improve* performance on target tasks, compared to fine-tuning.

4.4.4 Task Analogies

In this section, we explore task analogies in the form "A is to B as C is to D", and show that task arithmetic using vectors from the first three tasks improves performance on task D even if little or not data for that task is available.

Domain generalization. For many target tasks, gathering unlabeled data is easier and cheaper than collecting human annotations. When labeled data for a *target* task is not available, we can use task analogies to improve accuracy on the target task, using an *auxiliary* task for which there is labeled data and an unsupervised learning objective. For example, consider the target task of sentiment analysis using data from Yelp (Zhang et al., 2015). Using task analogies, we can construct a task vector $\hat{\tau}_{\text{yelp; sent}} = \tau_{\text{amazon; sent}} + (\tau_{\text{yelp; lm}} - \tau_{\text{amazon; lm}})$, where $\tau_{\text{amazon; sent}}$ is obtained by fine-tuning on labeled data from an auxiliary task (sentiment analysis using data from Amazon; McAuley and Leskovec (2013)), and

Table 4.11: **Improving domain generalization with task analogies.** Using an auxiliary task for which labeled data is available and unlabeled data from both the auxiliary and the target datasets, task analogies improve the accuracy for multiple T5 models and two sentiment analysis target tasks (Zhang et al., 2015; McAuley and Leskovec, 2013), without using any labeled data from the target tasks.

	target = Yelp			target = Amazon		
Method	T5-small	T5-base	T5-large	T5-small	T5-base	T5-large
Fine-tuned on auxiliary	88.6	92.3	95.0	87.9	90.8	94.8
Task analogies	89.9	93.0	95.1	89.0	92.7	95.2
Fine-tuned on target	91.1	93.4	95.5	90.2	93.2	95.5

$\tau_{\text{yelp; lm}}$ and $\tau_{\text{amazon; lm}}$ are task vectors obtained via (unsupervised) language modeling on the inputs from both datasets.

In Table 4.11, we show that using such task analogies improves accuracy of T5 models at multiple scales, both for Amazon and Yelp binary sentiment analysis as target tasks. We empirically found that giving a higher weight to the sentiment analysis task vector led to higher accuracy, and we thus used two independent scaling coefficients for these experiments—one for the sentiment analysis task vector and one for both the language modeling task vectors. Using task vectors outperforms fine-tuning on the remaining auxiliary sentiment analysis task for all models and datasets, approaching the performance of fine-tuning on the target task.

Subpopulations with little data. There is often some inherent scarcity in certain data subpopulations—for example, images of lions in indoor settings are more rare, compared to lions in outdoor settings or dogs in general (indoor or outdoors). Whenever such subpopulations admit analogies to others with more abundant data (as in this case), we can apply task analogies, e.g., $\hat{\tau}_{\text{lion indoors}} = \tau_{\text{lion outdoors}} + (\tau_{\text{dog indoors}} - \tau_{\text{dog outdoor}})$.

We explore this scenario by creating four subpopulations, using 125 overlapping classes

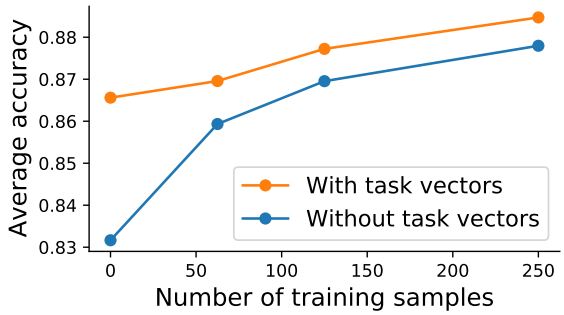

Figure 4.21: **Learning about subpopulations via analogy**. Combining task vectors from related subpopulations improves accuracy on the target subpopulation, when little or no data from the target supopulation is available. Accuracy is averaged over the four target subpopulations and three CLIP models.

between ImageNet and a dataset of human sketches (Eitz et al., 2012). We split these classes in two subsets of roughly equal size, creating four subpopulations A, B, C and D, where the pairs (A, C) and (B, D) share the same classes, and (A, B) and (C, D) share the same style (photo-realistic images or sketches). Although these subpopulations have many classes in our experiments, we use the simplified subsets "real dog", "real lion", "sketch dog" and "sketch lion" as a running example.

Given a target subpopulation, we create task vectors by fine-tuning three models independently on the remaining subpopulations, and then combine them via task arithmetic, e.g., $\hat{\tau}_{\text{sketch lion}} = \tau_{\text{sketch dog}} + (\tau_{\text{real lion}} - \tau_{\text{real dog}})$ for the target subpopulation "sketch lion". We show the results in Figure 4.21, averaged over the four target subpopulations. Compared to the pre-trained model, task vectors improve accuracy by 3.4 percentage points on average. Moreover, when some data from the target subpopulation is available for fine-tuning, starting from the edited model leads to consistently higher accuracy than starting from the pre-trained model. The gains from analogies alone (with no additional data) are roughly the same as that of collecting and annotating around one hundred training samples for the target subpopulation.

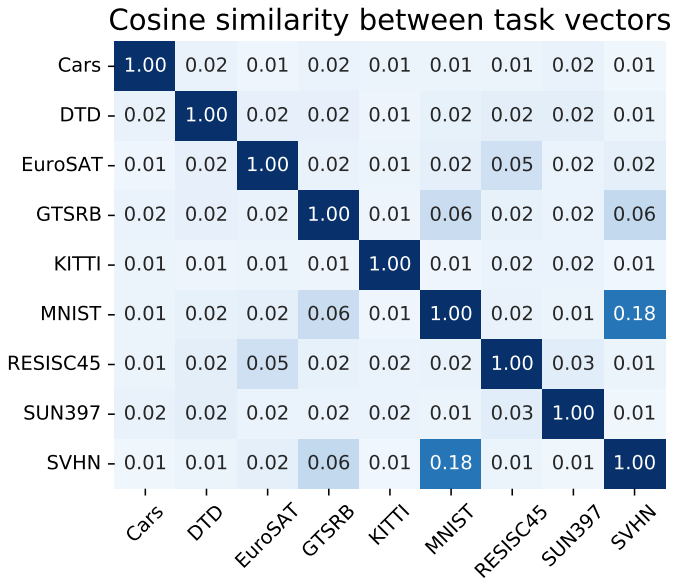

Figure 4.22: **Task vectors are typically close to orthogonal.** The plot shows the cosine similarities between vectors for different tasks, using CLIP. The largest deviations from orthogonality are found when tasks are similar to each other, for instance, for MNIST, SVHN and GTSRB—where recognizing digits is either the task itself (MNIST and SVHN), or a capability needed to solve the task (GTSRB, where the task is traffic sign recognition)—and EuroSAT and RESISC45, two satellite imagery recognition datasets.

4.4.5 Discussion

In this section, we provide further insight into previous results by exploring the similarity between task vectors for different tasks, as well as the impact of different learning rates and random seeds. We conclude by discussing some limitations of our approach.

Similarity between task vectors. In Figure 4.22, we explore the cosine similarity between task vectors for different tasks, in an effort to understand how multiple models can be collapsed into a single multi-task model via addition (Section 4.4.3). We observe that vectors from different tasks are typically close to orthogonal, and speculate that this enables the combination of task vectors via addition with minimal interference. We also observe higher cosine similarities when tasks are semantically similar to each other. For example,

the largest cosine similarities in Figure 4.22 (left) are between MNIST, SVHN and GTSRB, where recognizing digits is essential for the tasks, and between EuroSAT and RESISC45, which are both satellite imagery recognition datasets. This similarity in "task space" could help explain some results in Ilharco et al. (2022), where interpolating the weights of a model fine-tuned on one task and the pre-trained model weights—in our terminology, applying a single task vector—sometimes improves accuracy on a different task for which no data is available (e.g., applying the MNIST task vector improves accuracy on SVHN).

The impact of the learning rate. In Figure 4.23, we observe that increasing the learning rate degrades accuracy both when using task vectors and when fine-tuning individual models, but the decrease is more gradual for individual models. These findings align with those of (Wortsman et al., 2022a), who observed that accuracy decreases on the linear path between two fine-tuned models when using a larger learning rate. Thus, while larger learning rates may be acceptable when fine-tuning individual models, we recommend more caution when using task vectors. Further, we hypothesize that larger learning rates may explain some of the variance when adding vectors from natural language processing tasks, where we take models fine-tuned by others in the community.

The evolution of task vectors throughout fine-tuning. In Figure 4.24, we show how task vectors evolve throughout fine-tuning. Intermediate task vectors converge rapidly to the direction of the final task vector obtained at the end of fine-tuning. Moreover, the accuracy of the model obtained by adding intermediate task vectors from two image classification tasks saturates after just a few hundred steps. These results suggest that using intermediate task vectors can be a useful way of saving compute with little harm in accuracy.

Limitations. Task vectors are restricted to models with the same architecture, since they depend on element-wise operations on model weights. Further, in all of our experiments we perform arithmetic operations only on models fine-tuned from the same pre-trained initialization, although emerging work shows promise in relaxing this assumption (Ainsworth et al., 2022). We also note that some architectures are very popular, and have "standard" initializations—e.g., at the time of writing there are over 3,000 models on Hugging Face Hub fine-tuned from the same BERT-base initialization (Devlin et al., 2019a), and over 800

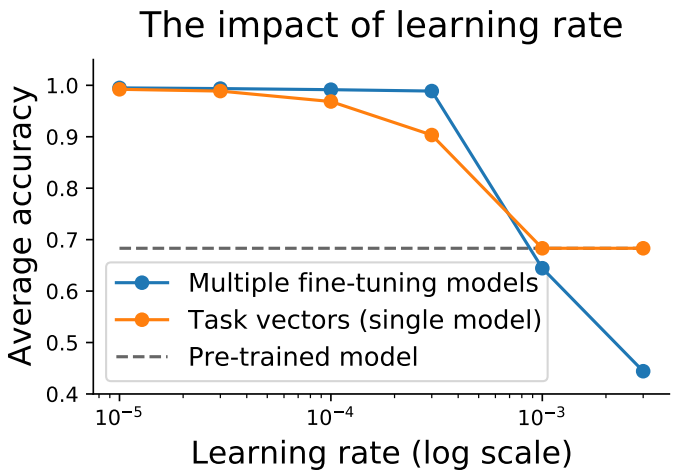

Figure 4.23: **The impact of learning rate when fine-tuning.** When adding task vectors from CLIP ViT-L/14 models fine-tuned on MNIST and EuroSAT, lower learning rates make the best use of the fine-tuned models, and also correspond to the highest accuracies of the fine-tuned models on the target task.

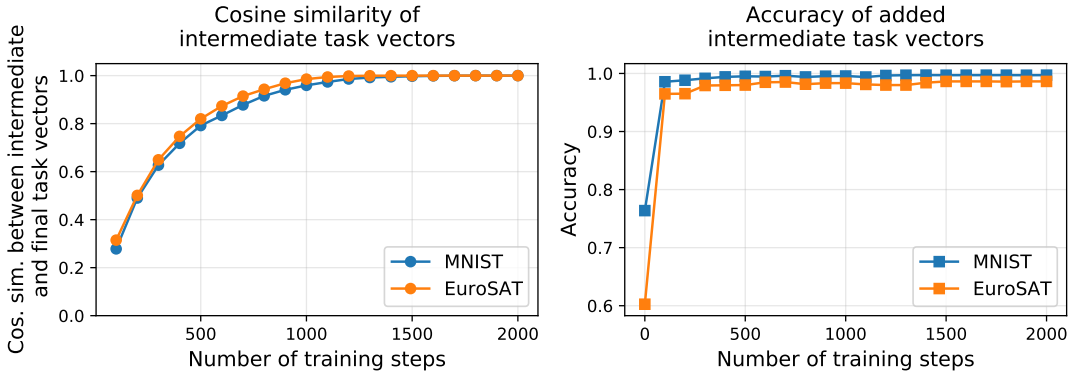

Figure 4.24: **How task vectors evolve throughout fine-tuning.** Left: the cosine similarity between the final task vector and task vectors produced at intermediate points during fine-tuning. Right: Accuracy obtained by adding intermediate task vectors from MNIST and EuroSAT. Adding intermediate task vectors can lead to high accuracy, despite fine-tuning for substantially fewer steps.

models fine-tuned from the same T5-small initialization.

4.4.6 Conclusion

In this section we introduce a new paradigm for editing models based on arithmetic operations over *task vectors*. For various vision and NLP models, *adding* multiple specialized task vectors results in a single model that performs well on all target tasks, or even improves performance on a single task. *Negating* task vectors allows users to remove undesirable behaviors, e.g., toxic generations, or even forget specific tasks altogether, while retaining performance everywhere else. Finally, *task analogies* leverage existing data to improve performance on domains or subpopulations where data is scarce.

Arithmetic operations over task vectors only involve adding or subtracting model weights, and thus are efficient to compute, especially when compared to alternatives that involve additional fine-tuning. Thus, users can easily experiment with various model edits, recycling and transferring knowledge from large collections of publicly available fine-tuned models. Since these operations result in a single model of the same size, they incur no extra inference cost. Our code is available at https://github.com/mlfoundations/task_vectors.

Chapter 5

FINAL CONSIDERATIONS

This dissertation focused on improving multimodal models, either from the ground up—through designing better datasets—or editing existing models—using efficient techniques that operate directly in a model's weight space. Together, these methods have allowed substantial progress in the capabilities of models like CLIP. One appealing aspect of these techniques is that they do not need to specialize the model for a specific purpose. In fact, our goal is to build models that are both general and reliable. A key insight towards that goal is to enable fine-tuning without sacrificing the flexible text-based interface that makes open-vocabulary models like CLIP so appealing.

The success of open-vocabulary multimodal models has arguably been enabled by their ability to perform well on a broad range of downstream tasks, even without fine-tuning them on specific tasks. This is possible thanks to the text-based interface that these models have, which allows specifying any classification problem in natural language. Flexible interfaces have been similarly impactful in natural language processing. Naturally, almost all NLP tasks can be seen as "text-to-text" problems, where models consume text as inputs and produce text as outputs (Brown et al., 2020a; Khashabi et al., 2020; Wang et al., 2022). Thanks to this interface, large language models like GPT-4 can be used for thousand of downstream tasks, without any adaptation (OpenAI, 2023).

A natural extension to this trend is to build models that have an even more flexible and general interface. At the extreme, this means building models that consume raw bytes as inputs and produce raw bytes as outputs. This interface makes it easy to see any machine learning task as a "byte-to-byte" problem which naturally suites the model's interface. Another advantage of such a system is that it would enable self-supervised learning from virtually any data source, including all media in the web. This is particularly appealing considering how models tend to improve with the scale of their pre-training data.

There are, however, some key technical challenges that need to be addressed before building such a model. The first challenge is designing architectures that are able to process very long sequences. The Transformer (Vaswani et al., 2017), one of the most popular architectures in recent years, scale poorly to long sequences due to a quadratic computational dependency in the length of the sequence. This makes it infeasible to use these models to process inputs such as high-resolution images at the byte level. While recent advances have been made in this direction (Tay et al., 2022; Yu et al., 2023; Chen and Li, 2023), building powerful architectures that can process very long sequences remains an open research problem.

Moreover, when building a multimodal byte-level model, there can be large asymmetries between the inputs of different modalities. For example, consider an image and it's corresponding alt-text on a web page. While the image might contain hundreds of thousands or even millions of bytes, the text is likely orders of magnitude shorter, with tens or hundreds of bytes. Moreover, the entropy of the bytes also depends highly on the modality. Yet, most current architectures are not designed to handle this asymmetry, spending the exact same amount of compute for all elements in the sequence. I believe designing architectures that can dynamically allocate compute to different parts of the input sequences will be key for building a byte-level multimodal model.

I believe tackling both of those challenges will be key for building the next generation of multimodal models. If progress is made in those fronts, it would enable training models with a highly flexible interface, that can be trained in an end-to-end fashion on virtually any collected data. I am excited about a future where our models can process and generate any form of data, and for the vast array of applications that this interface would make possible.

BIBLIOGRAPHY

Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication, 2023. https://arxiv.org/abs/2303.09540.

Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *Journal of the ACM (JACM)*, 2004. https://doi.org/10.1145/1008731.1008736.

Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *European conference on computer vision*, pages 329–344. Springer, 2014.

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries, 2022. https://arxiv.org/abs/2209.04836.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. https://openreview.net/forum?id=EbMuimAbPbs.

Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4845–4854, 2019.

Anders Andreassen, Yasaman Bahri, Behnam Neyshabur, and Rebecca Roelofs. The evolution of out-of-distribution robustness throughout fine-tuning, 2021. https://arxiv.org/abs/2106.15831.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015. https://arxiv.org/abs/1505.00468.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment, 2021. https://arxiv.org/abs/2112.00861.

Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 36–45, 2015.

Olivier Bachem, Mario Lucic, and Andreas Krause. Coresets for nonparametric estimation - the case of dp-means. In *International Conference on Machine Learning (ICML)*, 2015. https://proceedings.mlr.press/v37/bachem15.html.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *II PASCAL challenge*, 2006.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/97af07a14cacba681feacf3012730892-Paper.pdf.

Sara Beery, Arushi Agarwal, Elijah Cole, and Vighnesh Birodkar. The iwildcam 2021 competition dataset. *arXiv preprint arXiv:2105.03494*, 2021.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal rec-

ognizing textual entailment challenge. In *TAC*, 2009. https://cris.fbk.eu/handle/11582/5351.

Gregory Benton, Wesley Maddox, Sanae Lotfi, and Andrew Gordon Gordon Wilson. Loss surface simplexes for mode connecting volumes and fast ensembling. In *International Conference on Machine Learning (ICML)*, 2021. https://arxiv.org/abs/2102.13042.

Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 2018. https://arxiv.org/abs/1712.03141.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, 2013. https://arxiv.org/abs/1708.06131.

Vighnesh Birodkar, Hossein Mobahi, and Samy Bengio. Semantic redundancies in image-classification datasets: The 10% you don't need. *arXiv preprint arXiv:1901.11409*, 2019. https://arxiv.org/abs/1901.11409.

Yonatan Bitton, Nitzan Bitton Guetta, Ron Yosef, Yuval Elovici, Mohit Bansal, Gabriel Stanovsky, and Roy Schwartz. WinoGAViL: Gamified association benchmark to challenge vision-and-language models, 2022. https://arxiv.org/abs/2207.12576.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models, 2021. https://arxiv.org/abs/2108.07258.

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of the 2019 World Wide Web Conference*, 2019. https://arxiv.org/abs/1903.04561.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)*, 2014. https://link.springer.com/chapter/10.1007/978-3-319-10599-4_29.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2020a. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020b. https://arxiv.org/abs/2005.14165.

Minwoo Byeon, Beomhee Park, Haecheon Kim, Sungjun Lee, Woonhyuk Baek, and Saehoon Kim. Coyo-700m: Image-text pair dataset. https://github.com/kakaobrain/coyo-dataset, 2022.

Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical, 2023. https://arxiv.org/abs/2302.10149.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments, 2020. https://arxiv.org/abs/2006.09882.

Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. https://arxiv.org/abs/2102.08981.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

Ting Chen and Lala Li. Fit: Far-reaching interleaved transformers. *arXiv preprint arXiv:2305.12689*, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners, 2020b. https://arxiv.org/abs/2006.10029.

Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish Thapliyal, James Bradbury, Weicheng Kuo, Mojtaba Seyedhosseini, Chao Jia, Burcu Karagol Ayan, Carlos Riquelme, Andreas Steiner, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. Pali: A jointly-scaled multilingual language-image model. In *International Conference on Learning Representations (ICLR)*, 2022. https://arxiv.org/abs/2209.06794.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. corr abs/2011.10566 (2020), 2020. https://arxiv.org/abs/2011.10566.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server, 2015. https://arxiv.org/abs/1504.00325.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)*, 2017. https://ieeexplore.ieee.org/abstract/document/7891544.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning, 2022. https://arxiv.org/abs/2212.07143.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. https://proceedings.neurips.cc/paper/2019/file/ae614c557843b1df326cb29c57225459-Paper.pdf.

Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. Fusing finetuned models for better pretraining, 2022. https://arxiv.org/abs/2204.03044.

Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. https://arxiv.org/abs/1711.07846.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. https://openaccess.thecvf.com/content_cvpr_2014/html/Cimpoi_Describing_Textures_in_2014_CVPR_paper.html.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. https://proceedings.mlr.press/v15/coates11a.html.

Michael B. Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *ACM-SIAM Symposium on Discrete Algorithms*, 2017. https://dl.acm.org/doi/10.5555/3039686.3039801.

C Coleman, C Yeh, S Mussmann, B Mirzasoleiman, P Bailis, P Liang, J Leskovec, and M Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations (ICLR)*, 2020. https://arxiv.org/abs/1906.11829.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

Wojciech Marian Czarnecki, Simon Osindero, Razvan Pascanu, and Max Jaderberg. A deep neural network's loss surface contains every low-dimensional pattern, 2019. https://arxiv.org/abs/1912.07559.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, 2005. https://link.springer.com/chapter/10.1007/11736790_9.

Zihang Dai, Hanxiao Liu, Quoc Le, and Mingxing Tan. CoAtNet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34, 2021.

Achal Dave, Tarasha Khurana, Pavel Tokmakov, Cordelia Schmid, and Deva Ramanan. Tao: A large-scale benchmark for tracking any object. In *European Conference on Computer Vision (ECCV)*, 2020. https://arxiv.org/abs/2005.10356.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021. https://arxiv.org/abs/2104.08164.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. https://ieeexplore.ieee.org/abstract/document/5206848.

Karan Desai, Gaurav Kaul, Zubin Aysola, and Justin Johnson. Redcaps: Web-curated image-text data created by the people, for the people, 2021. https://arxiv.org/abs/2111.11431.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019a. https://aclanthology.org/N19-1423.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding, June 2019b. URL https://aclanthology.org/N19-1423.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

Josip Djolonga, Jessica Yung, Michael Tschannen, Rob Romijnders, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Matthias Minderer, Alexander D'Amour, Dan Moldovan, Sylvain Gelly, Neil Houlsby, Xiaohua Zhai, and Mario Lucic. On robustness and transferability of convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. https://arxiv.org/abs/2007.08558.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping, 2020. https://arxiv.org/abs/2002.06305/.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *International Workshop on Paraphrasing*, 2005. https://aclanthology.org/I05-5002.

Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Cold fusion: Collaborative descent for distributed multitask finetuning, 2022. https://arxiv.org/abs/2212.01378.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. https://openreview.net/forum?id=YicbFdNTTy.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning (ICML)*, 2018. https://arxiv.org/abs/1803.00885.

Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on graphics (TOG)*, 2012. https://dl.acm.org/doi/10.1145/2185520.2185540.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations (ICLR)*, 2022. https://arxiv.org/abs/2110.06296.

Christian Ertler, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, Gerhard Neuhold, and Yubin Kuang. The mapillary traffic sign dataset for detection and classification on a global scale. In *European Conference on Computer Vision (ECCV)*, 2020. https://arxiv.org/abs/1909.04422.

Sabri Eyuboglu, Bojan Karlaš, Christopher Ré, Ce Zhang, and James Zou. dcbench: a benchmark for data-centric ai systems. In *Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning*, 2022. https://dl.acm.org/doi/abs/10.1145/3533028.3533310.

Alex Fang, Gabriel Ilharco, Mitchell Wortsman, Yuhao Wan, Vaishaal Shankar, Achal Dave, and Ludwig Schmidt. Data determines distributional robustness in contrastive language

image pre-training (clip). In *International Conference on Machine Learning (ICML)*, 2022. https://arxiv.org/abs/2205.01397.

Dan Feldman, Matthew Faulkner, and Andreas Krause. Scalable training of mixture models via coresets. In *Advances in Neural Information Processing Systems (NeuIPS)*, 2011. https://proceedings.neurips.cc/paper_files/paper/2011/file/2b6d65b9a9445c4271ab9076ead5605a-Paper.pdf.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. https://openreview.net/forum?id=6Tm1mposlrM.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective, 2019. https://arxiv.org/abs/1912.02757.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. https://arxiv.org/abs/2010.15110.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020. https://proceedings.mlr.press/v119/frankle20a.html.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 1999. https://www.sciencedirect.com/science/article/pii/S1364661399012942.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1997. https://www.sciencedirect.com/science/article/pii/S002200009791504X.

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, et al. Datacomp: In search of the next generation of multimodal datasets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. https://arxiv.org/abs/2304.14108.

Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters, 2021. https://arxiv.org/abs/2110.04544.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. https://arxiv.org/abs/1802.10026.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. https://ieeexplore.ieee.org/abstract/document/6248074.

Robert Geirhos, Carlos R Medina Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Generalisation in humans and deep neural networks. *arXiv preprint arXiv:1808.08750*, 2018.

Mor Geva, Avi Caciularu, Guy Dar, Paul Roit, Shoval Sadde, Micah Shlain, Bar Tamir, and Yoav Goldberg. Lm-debugger: An interactive tool for inspection and intervention in transformer-based language models, 2022. https://arxiv.org/abs/2204.12130.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third pascal recognizing textual entailment challenge. In *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 2007. https://aclanthology.org/W07-1401/.

Amelia Glaese, Nat McAleese, Maja Trebacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias,

Richard Green, Sona Mokra, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. Improving alignment of dialogue agents via targeted human judgements, 2022. https://www.deepmind.com/blog/building-safer-dialogue-agents.

Karan Goel, Albert Gu, Yixuan Li, and Christopher Ré. Model patching: Closing the subgroup performance gap with data augmentation, 2020. https://arxiv.org/abs/2008.06775.

Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. https://distill.pub/2021/multimodal-neurons.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. https://arxiv.org/abs/1911.04933.

Raphael Gontijo-Lopes, Yann Dauphin, and Ekin D Cubuk. No one representation to rule them all: Overlapping features of training methods, 2021. https://arxiv.org/abs/2007.01434.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning, 2022. https://arxiv.org/abs/2204.08499.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

Jia Guo, Jiankang Deng, Alexandros Lattas, and Stefanos Zafeiriou. Sample and computation redistribution for efficient face detection. In *International Conference on Learning Representations (ICLR)*, 2021. https://arxiv.org/abs/2105.04714.

Laura Hanu and Unitary team. Detoxify, 2020. https://github.com/unitaryai/detoxify.

Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Symposium on Theory of Computing (STOC)*, 2004. https://doi.org/10.1145/1007352.1007400.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. https://arxiv.org/abs/1512.03385.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. https://arxiv.org/abs/1709.00029.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations (ICLR)*, 2019.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021a. https://arxiv.org/abs/2006.16241.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021b.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021c. https://arxiv.org/abs/1907.07174.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models, 2022a. https://arxiv.org/abs/2203.15556.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models, 2022b. https://arxiv.org/abs/2203.15556.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. OpenCLIP, July 2021. https://doi.org/10.5281/zenodo.5143773.

Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. https://arXiv.org/abs/2208.05592.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023. https://arxiv.org/abs/2212.04089.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. https://arxiv.org/abs/1803.05407.

Tanuj Jain, Christopher Lennan, Zubin John, and Dat Tran. Imagededup, 2019. https://github.com/idealo/imagededup.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning (ICML)*, 2021. https://arxiv.org/abs/2102.05918.

Mon-Fong Jiang, Shian-Shyong Tseng, and Chih-Ming Su. Two-phase clustering process

for outliers detection. *Pattern recognition letters*, 2001. https://www.sciencedirect.com/science/article/abs/pii/S0167865500001318.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 2019. https://arxiv.org/abs/1702.08734.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. https://arxiv.org/abs/1612.06890.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2017. https://arxiv.org/abs/1607.01759.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Atoosa Kasirzadeh and Iason Gabriel. In conversation with artificial intelligence: aligning language models with human values, 2022. https://arxiv.org/abs/2209.00731.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics (EMNLP)*, 2020. https://aclanthology.org/2020.findings-emnlp.171.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 2017. https://arxiv.org/abs/1612.00796.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony

Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, 2021. https://arxiv.org/abs/2012.07421.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning (ICML)*, 2019a. https://arxiv.org/abs/1905.00414.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019b. https://arxiv.org/abs/1805.08974.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *International Conference on Computer Vision Workshops (ICML)*, 2013. https://www.cv-foundation.org/openaccess/content_iccv_workshops_2013/W19/html/Krause_3D_Object_Representations_2013_ICCV_paper.html.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

Rohith Kuditipudi, Xiang Wang, Holden Lee, Yi Zhang, Zhiyuan Li, Wei Hu, Rong Ge, and Sanjeev Arora. Explaining landscape connectivity of low-cost solutions for multilayer nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. https://arxiv.org/abs/1906.06247.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *Interna-*

tional Conference on Learning Representations (ICLR), 2022a. https://openreview.net/forum?id=UYneFzXSJWh.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=UYneFzXSJWh.

Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

Yann LeCun. The MNIST database of handwritten digits, 1998. http://yann.lecun.com/exdb/mnist/.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021. https://arxiv.org/abs/2107.06499.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. https://arxiv.org/abs/1703.08475.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss

landscape of neural nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. https://arxiv.org/abs/1712.09913.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models, 2022. https://arxiv.org/abs/2208.03306.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021. https://aclanthology.org/2021.acl-long.522.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Shangyun Lu, Bradley Nott, Aaron Olson, Alberto Todeschini, Hossein Vahabi, Yair Carmon, and Ludwig Schmidt. Harder or different? a closer look at distribution shift in dataset reproduction. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2020.

Ximing Lu, Sean Welleck, Liwei Jiang, Jack Hessel, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. Quark: Controllable text generation with reinforced unlearning, 2022. https://arxiv.org/abs/2205.13636.

Ekdeep Singh Lubana, Eric J Bigelow, Robert P Dick, David Krueger, and Hidenori Tanaka. Mechanistic mode connectivity, 2022. https://arxiv.org/abs/2211.08422.

Mario Lucic, Matthew Faulkner, Andreas Krause, and Dan Feldman. Training gaussian mixture models at scale via coresets. *Journal of Machine Learning Research (JMLR)*, 2018. http://jmlr.org/papers/v18/15-506.html.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. https://arxiv.org/abs/2111.09832.

Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 1975. https://www.sciencedirect.com/science/article/abs/pii/0005279575901099.

Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Douwe Kiela, David Jurado, David Kanter, Rafael Mosquera, Juan Ciro, Lora Aroyo, Bilge Acun, Sabri Eyuboglu, Amirata Ghorbani, Emmett Goodman, Tariq Kane, Christine R. Kirkpatrick, Tzu-Sheng Kuo, Jonas Mueller, Tristan Thrush, Joaquin Vanschoren, Margaret Warren, Adina Williams, Serena Yeung, Newsha Ardalani, Praveen Paritosh, Ce Zhang, James Zou, Carole-Jean Wu, Cody Coleman, Andrew Ng, Peter Mattson, and Vijay Janapa Reddi. Dataperf: Benchmarks for data-centric ai development, 2022. https://arxiv.org/abs/2207.10062.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM Conference on Recommender Systems*, 2013. https://dl.acm.org/doi/10.1145/2507157.2507163.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*. Elsevier, 1989. https://www.sciencedirect.com/science/article/abs/pii/S0079742108605368.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. https://arxiv.org/abs/1609.07843.

George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, Nov 1995. https://doi.org/10.1145/219717.219748.

John P Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In Marina Meila and Tong Zhang, editors, *International Conference on Machine Learning*

(ICML), volume 139 of *Proceedings of Machine Learning Research*, pages 7721–7735. PMLR, 18–24 Jul 2021. URL http://proceedings.mlr.press/v139/miller21b.html.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning (ICML)*, 2020. https://arxiv.org/abs/1906.01827.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*, 2021. https://arxiv.org/abs/2110.11309.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, 2022. https://arxiv.org/abs/2206.06520.

Norman Mu, Alexander Kirillov, David Wagner, and Saining Xie. Slip: Self-supervision meets language-image pre-training, 2021. https://arxiv.org/abs/2112.12750.

Shikhar Murty, Christopher D Manning, Scott Lundberg, and Marco Tulio Ribeiro. Fixing model bugs with natural language patches. In *ACL Workshop on Learning with Natural Language Supervision*, 2022. https://openreview.net/forum?id=blJrg3WvvDV.

Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *NeurIPS*, 2019. https://proceedings.neurips.cc/paper/2019/file/05e97c207235d63ceb1db43c60db7bbb-Paper.pdf.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2011. https://storage.googleapis.com/pub-tools-public-publication-data/pdf/37648.pdf.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. https://arxiv.org/abs/2008.11687.

Andrew Ng, Dillon Laird, and Lynn He. Data-centric ai competition, 2021. https://https-deeplearning-ai.github.io/data-centric-comp/.

Thao Nguyen, Gabriel Ilharco, Mitchell Wortsman, Sewoong Oh, and Ludwig Schmidt. Quality not quantity: On the interaction between dataset design and robustness of clip. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. https://openreview.net/forum?id=LTCBavFWp5C.

OpenAI. Gpt-4 technical report, 2023. https://arxiv.org/abs/2303.08774.

Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. Im2text: Describing images using 1 million captioned photographs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011. https://papers.nips.cc/paper_files/paper/2011/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback, 2022. https://arxiv.org/abs/2203.02155.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. https://arxiv.org/abs/1912.01703.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. https://arxiv.org/abs/2107.07075.

Hieu Pham, Zihang Dai, Golnaz Ghiasi, Hanxiao Liu, Adams Wei Yu, Minh-Thang Luong, Mingxing Tan, and Quoc V Le. Combined scaling for zero-shot transfer learning, 2021a. https://arxiv.org/abs/2111.10050.

Hieu Pham, Zihang Dai, Golnaz Ghiasi, Hanxiao Liu, Adams Wei Yu, Minh-Thang Luong,

Mingxing Tan, and Quoc V. Le. Combined scaling for zero-shot transfer learning, 2021b. https://arxiv.org/abs/2111.10050.

Boris Teodorovich Polyak. New method of stochastic approximation type. *Automation and remote control*, 1990.

Filip Radenovic, Abhimanyu Dubey, Abhishek Kadian, Todor Mihaylov, Simon Vandenhende, Yash Patel, Yi Wen, Vignesh Ramanathan, and Dhruv Mahajan. Filtering, distillation, and hard negatives for vision-language pre-training. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. https://arxiv.org/abs/2301.02280.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners, 2019. https://openai.com/blog/better-language-models/.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. https://arxiv.org/abs/2103.00020.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

Colin Raffel. A call to build models like we build open-source software, 2021. https://colinraffel.com/blog/a-call-to-build-models-like-we-build-open-source-software.html.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 2020a. http://jmlr.org/papers/v21/20-074.html.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research (JMLR)*, 2020b. https://arxiv.org/abs/1910.10683.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140): 1–67, 2020c. URL http://jmlr.org/papers/v21/20-074.html.

Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations (ICLR)*, 2021. https://openreview.net/forum?id=GhVS8_yPeEa.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning (ICML)*, 2021. https://arxiv.org/abs/2102.12092.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. https://arxiv.org/abs/2204.06125.

J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad, 2018.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning (ICML)*, 2019a. http://proceedings.mlr.press/v97/recht19a.html.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5389–5400. PMLR, 09–15 Jun 2019b. URL http://proceedings.mlr.press/v97/recht19a.html.

Marco Tulio Ribeiro and Scott Lundberg. Adaptive testing and debugging of nlp models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022. https://aclanthology.org/2022.acl-long.230/.

Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C Mozer. Mitigating bias in calibration error estimation, 2020. https://arxiv.org/abs/2012.08668.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. https://arxiv.org/abs/2112.10752.

Peter J Rousseeuw and Mia Hubert. Robust statistics for outlier detection. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, 2011. http://i2pc.es/coss/Docencia/SignalProcessingReviews/Rousseeuw2011.pdf.

Peter J Rousseeuw and Mia Hubert. Anomaly detection by robust statistics. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, 2018. https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1236.

David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015. https://arxiv.org/abs/1409.0575.

Shiori Sagawa, Pang Wei Koh, Tony Lee, Irena Gao, Sang Michael Xie, Kendrick Shen, Ananya Kumar, Weihua Hu, Michihiro Yasunaga, Henrik Marklund, Sara Beery, Etienne David, Ian Stavness, Wei Guo, Jure Leskovec, Kate Saenko, Tatsunori Hashimoto, Sergey Levine, Chelsea Finn, and Percy Liang. Extending the wilds benchmark for unsupervised adaptation. In *International Conference on Learning Representations (ICLR)*, 2022. https://arxiv.org/abs/2112.05090.

Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. https://arxiv.org/abs/2112.01008.

Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. LAION-400M: Open dataset of clip-filtered 400 million image-text pairs, 2021. https://arxiv.org/abs/2111.02114.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022. https://openreview.net/forum?id=M3Y74vmsMcY.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations (ICLR)*, 2018. https://openreview.net/forum?id=H1aIuk-RW.

Vaishaal Shankar, Achal Dave, Rebecca Roelofs, Deva Ramanan, Benjamin Recht, and Ludwig Schmidt. Do image classifiers generalize across time? *arXiv preprint arXiv:1906.02168*, 2019.

Vaishaal Shankar, Rebecca Roelofs, Horia Mania, Alex Fang, Benjamin Recht, and Ludwig Schmidt. Evaluating machine accuracy on imagenet. In *International Conference on Machine Learning (ICML)*, 2020. http://proceedings.mlr.press/v119/shankar20c/shankar20c.pdf.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image caption-

ing. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018. https://aclanthology.org/P18-1238/.

Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. How much can clip benefit vision-and-language tasks? In *International Conference on Learning Representations (ICLR)*, 2022. https://arxiv.org/abs/2107.06383.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013. https://aclanthology.org/D13-1170/.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. https://openreview.net/forum?id=UmvSlP-PyV.

Krishna Srinivasan, Karthik Raman, Jiecao Chen, Michael Bendersky, and Marc Najork. Wit: Wikipedia-based image text dataset for multimodal multilingual machine learning. In *44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021. https://arxiv.org/abs/2103.01913.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *International Joint Conference on Neural Networks (IJCNN)*, 2011. https://ieeexplore.ieee.org/document/6033395.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics (ACL)*, 2022. https://aclanthology.org/2022.findings-acl.48.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreason-

able effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. https://arxiv.org/abs/2111.09839.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. https://arxiv.org/abs/2112.06825.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition*, 2016. https://arxiv.org/abs/1512.00567v3.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. https://dl.acm.org/doi/abs/10.5555/3495724.3497285.

Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning, 2021. https://arxiv.org/abs/2111.08947.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28, 2022.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 2016a. https://arxiv.org/abs/1503.01817.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 2016b. https://arxiv.org/abs/1503.01817.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, 1998. https://link.springer.com/chapter/10.1007/978-1-4615-5529-2_8.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations (ICLR)*, 2018. https://arxiv.org/abs/1812.05159.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. https://arxiv.org/abs/1706.03762.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*, 2018. https://arxiv.org/abs/1804.07461.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, 2022.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics (TACL)*, 2019. https://aclanthology.org/Q19-1040/.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active

learning. In *International Conference on Machine Learning (ICML)*, 2015. https://proceedings.mlr.press/v37/wei15.html.

Mitchell Wortsman, Maxwell C Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. Learning neural network subspaces. In *International Conference on Machine Learning (ICML)*, 2021. http://proceedings.mlr.press/v139/wortsman21a.html.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022a. https://arxiv.org/abs/2203.05482.

Mitchell Wortsman, Gabriel Ilharco, Mike Li, Jong Wook Kim, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022b. https://arxiv.org/abs/2109.01903.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017. https://arxiv.org/abs/1708.07747.

Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision (IJCV)*, 2016. https://link.springer.com/article/10.1007/s11263-014-0748-y.

Kaiyu Yang, Jacqueline H Yau, Li Fei-Fei, Jia Deng, and Olga Russakovsky. A study of face obfuscation in ImageNet. In *International Conference on Machine Learning (ICML)*, 2022. https://arxiv.org/abs/2103.06191.

Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. Filip: Fine-grained interactive language-image pre-training. In *International Conference on Learning Representations (ICLR)*, 2022. https://arxiv.org/abs/2111.07783.

Shuhei Yokoo. Contrastive learning with large memory bank and negative embedding subtraction for accurate copy detection, 2021. https://arxiv.org/abs/2112.04323.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2014. https://aclanthology.org/Q14-1006/.

Dantong Yu, Gholamhosein Sheikholeslami, and Aidong Zhang. Findout: Finding outliers in very large datasets. *Knowledge and information Systems*, 2002. https://link.springer.com/article/10.1007/s101150200013.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022. https://arxiv.org/abs/2205.01917.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. Megabyte: Predicting million-byte sequences with multiscale transformers. *arXiv preprint arXiv:2305.07185*, 2023.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision, 2021. https://arxiv.org/abs/2111.11432.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 2016. https://dl.acm.org/doi/10.1145/2934664.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, André Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. The visual task adaptation benchmark, 2019. http://arxiv.org/abs/1910.04867.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *arXiv preprint arXiv:2106.04560*, 2021.

Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. https://arxiv.org/abs/2111.07991.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2017. https://arxiv.org/abs/1710.09412.

Renrui Zhang, Rongyao Fang, Peng Gao, Wei Zhang, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling, 2021. https://arxiv.org/abs/2111.03930.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. https://arxiv.org/abs/2203.05557.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying memories in transformer models, 2020. https://arxiv.org/abs/2012.00363.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 2020. https://arxiv.org/abs/1911.02685.