UNIVERSITY OF CALIFORNIA

Los Angeles

Data-efficient and Robust Deep Learning from Large Vision and Language Data

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Yu Yang

2024

ABSTRACT

Data-efficient and Robust Deep Learning from Large Vision and Language Data

by

Yu Yang
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Baharan Mirzasoleiman, Chair

Deep learning has revolutionized fields like computer vision, natural language processing, and multimodal learning, but its reliance on large datasets brings challenges such as rising computational costs, vulnerability to data poisoning attacks, and difficulty achieving robustness against spurious correlations.

My research addresses these challenges through a data-centric approach, improving data selection, curriculum design, and weighting strategies. This dissertation is organized into three parts. First, for efficient training, CREST identifies coresets for deep vision models with theoretical guarantees, and S2L reduces fine-tuning costs for large language models by prioritizing subsets based on proxy model loss trajectories. Second, for robust training against data poisoning, EPIC iteratively detects and excludes malicious examples during training, effectively mitigating the attacks. Finally, to address spurious correlations, SPARE mitigates these biases early in training by separating and rebalancing biased groups, PDE progressively expands balanced subsets to guide models toward learning core features, and a multimodal fine-tuning method enhances robustness in vision-language models like CLIP by reducing reliance on spurious features, achieving significant gains in worst-group accuracy.

Together, my research demonstrates how focusing on the properties and selection of data helps address core limitations in deep learning, providing scalable and effective solutions that bridge theoretical insights with practical needs across diverse real-world applications.

The dissertation of Yu Yang is approved.

Aditya Grover

Quanquan Gu

Guy Van den Broeck

Baharan Mirzasoleiman, Committee Chair

University of California, Los Angeles

2024

Table of Contents

List of Figures . . . x

Acknowledgments . . . xviii

Curriculum Vitae . . . xxii

1 Introduction . . . 1

1.1 Research Contributions . . . 3

1.2 Real-world Impacts . . . 4

I Data Selection for Efficient Training 5

2 CREST: Data-efficient Training for Deep Vision Models . . . 6

2.1 Related Work . . . 8

2.2 Problem Formulation and Background . . . 9

2.3 Coresets for Training Non-convex Models . . . 11

2.3.1 Modeling the Non-convex Loss Function . . . 13

2.3.2 Coresets for (Mini-batch) Stochastic GD . . . 16

2.3.3 Further Improving Efficiency of Coreset Selection . . . 18

2.4 Experiments . . . 21

2.4.1 Evaluating Accuracy and Speedup . . . 22

2.4.2 Ablation Study . . . 23

2.5 Conclusion . . . 26

3 S2L: Data-Efficient Training for Large Language Models . . . 28

3.1 Related Work 30

3.2 Problem Formulation 32

3.3 Methodology 33

3.4 Experiments 36

3.4.1 Baselines 36

3.4.2 Specialized Domain 1: Mathematical Reasoning 37

3.4.3 Specialized Domain 2: Clinical Text Summarization 41

3.4.4 Ablation Studies 43

3.5 Conclusion and Limitations 45

II Data Selection for Robust Training against Data Poisoning 48

4 EPIC: Robust Training Against Data Poisoning 49

4.1 Related Work 51

4.1.1 Targeted Data Poisoning 51

4.1.2 Defense Strategies 52

4.2 Robust Training against Data Poisoning 53

4.2.1 Motivation 54

4.2.2 Not all the poisons are created equal 55

4.2.3 Effective poisons are not examples with highest loss or lowest confidence 57

4.2.4 Effective poisons become isolated in gradient space 57

4.2.5 Eliminating the effective poisons 58

4.2.6 Adaptive attacks 62

4.3 Experiments 63

4.3.1 Against Data Poisoning Attacks 63

4.3.2 Comparison to SOTA Defenses against GM 67

4.3.3 Comparison under Larger Perturbations 68

4.4 Conclusion 69

III Data-efficient and Robust Training against Spurious Correlations 70

5 SPARE: Identifying Spurious Correlations Early in Training 71

5.1 Related Work 73

5.2 Problem Formulation 74

5.3 Investigating How Spurious Features are Learned by Neural Networks 76

5.3.1 Spurious Features are Learned in the Initial Training Iterations . . . 77

5.3.2 Network Exclusively Relies on Simple Spurious Features on Majority of Examples 79

5.4 Eliminating Spurious Bias Early in Training 81

5.5 Experiments 82

5.5.1 Mitigating Spurious Correlations in Benchmark Datasets 83

5.5.2 Ablation Studies 86

5.5.3 Discovering and Mitigating Spurious Correlations in Restricted ImageNet 87

5.6 Conclusion 88

6 PDE: Data-efficient and Robust Training against Spurious Correlations 90

6.1 Why is Spurious Correlation Harmful to ERM? 92

6.1.1 Empirical Risk Minimization 93

6.1.2 Data Distribution with Spurious Correlation Fails ERM 93

6.1.3 Beyond Linear Models 95

6.1.4 Understanding the Training Process with Spurious Correlation 95

6.2 Theory-Inspired Two-Stage Training Algorithm 97

6.2.1 Theoretical Implications 97

6.2.2 PDE: A Two-Stage Training Algorithm 98

6.3 Experiments 101

6.3.1 Synthetic Data 101

6.3.2 Real Data 102

6.4 Related Work 107

6.5 Conclusion 108

7 Fine-tuning against Spurious Correlations for Vision-Language Models . 110

7.1 Related Work 113

7.2 Spurious-aware Contrastive Language Image Fine-tuning 115

7.3 Spurious Correlation Detection 118

7.3.1 Methodology 119

7.4 Experiments 120

7.4.1 Backbones 120

7.4.2 Datasets 121

7.4.3 Metrics 122

7.4.4 Baselines 124

7.4.5 Spurious Correlation Detection Results 125

7.4.6 Spurious Correlation Mitigation Results 126

7.5 Conclusion and Future Work 128

8 Conclusion and Future Work 133

9 Appendices . . . 136

9.1 Appendix for Chapter 2 . . . 136

9.1.1 Proofs . . . 136

9.1.2 Experimental details . . . 143

9.2 Appendix for Chapter 3 . . . 147

9.2.1 Proofs . . . 147

9.2.2 Experiment Details . . . 150

9.2.3 Examples in Different Clusters . . . 157

9.2.4 Topic Distribution of Data Selected by SPARE . . . 158

9.2.5 Broader Impacts . . . 158

9.3 Appendix for Chapter 4 . . . 159

9.3.1 Proof of Theorem 4.2.1 . . . 159

9.4 Appendix for Chapter 5 . . . 163

9.4.1 Simplicity Bias . . . 163

9.4.2 Setting . . . 163

9.4.3 Proof for Theorems . . . 166

9.4.4 Experimentation Details . . . 175

9.4.5 Discovering Spurious Features . . . 177

9.4.6 Comparing Inferred with Ground-truth Groups . . . 180

9.4.7 Reproducibility . . . 182

9.5 Appendix for Chapter 6 . . . 182

9.5.1 Synthetic Experiments . . . 182

9.5.2 Benchmark Datasets . . . 186

9.5.3 Real Data Experiments . . . 186

9.5.4 Proof Preliminaries 189

9.5.5 Proof of Theorem 6.1.2 191

9.5.6 Proof of Lemma 6.2.1 197

9.5.7 Auxiliary Lemmas 198

9.6 Appendix for Chapter 7 200

9.6.1 Ablation Study 200

9.6.2 Limitations 200

List of Figures

2.1 Training ResNet20 on CIFAR-10. (a) 10% Craig coresets selected at the beginning of every epoch from full data may perform very poorly. This is because, **(C1)**: (b) Coresets may have a large error: $\|\mathbf{g}_{t,S} - \nabla\mathcal{L}(\boldsymbol{w}_t)\|$, after a few training iterations; and **(C2)**: Gradient of weighted mini-batches selected from the coresets may have a (c) large bias $\|\mathbb{E}_i[\mathbf{g}_{t,\mathcal{M}_i}] - \nabla\mathcal{L}(\boldsymbol{w}_t)\|$ and (d) large variance $\mathbb{E}_i[\|\mathbf{g}_{t,\mathcal{M}_i} - \nabla\mathcal{L}(\boldsymbol{w}_t)\|^2]$, where $\mathcal{M}_i \in S$ is a mini-batch and $\mathbf{g}_{t,\mathcal{M}_i} = \mathbb{E}_{j\in\mathcal{M}_i}[\gamma_j \mathbf{g}_{t,j}]$. In contrast, our Crest coresets are nearly unbiased, and have a smaller variance than random mini-batches of same size. 12

2.2 Normalized run-time and test accuracy of Crest by that of full data, when training ResNet-20 on CIFAR10, ResNet-18 on CIFAR100, ResNet-50 on TinyImagenet, and fine-tuning RoBERTa on SNLI. 22

2.3 Normalized test accuracy and number of coreset updates for Crest over greedily selecting every mini-batch from a larger random subset by solving Eq. (2.5). . . 24

2.4 Training ResNet-20 on CIFAR-10 with Crest under 10% training budget. (Left) Number of coreset updates vs. training iterations. (Right) test accuracy vs. the total number of coreset updates. 25

2.5 Average forgettability score of Crest coresets during training, when learned examples are not discarded (Left), and are discarded (Right). Learning difficulty of examples selected by Crest increases during the training. 26

3.1 Existing data selection methods depend heavily on the feature representations from a reference model, which makes their effectiveness vulnerable to the quality of training on the target domain [118]. For supervised fine-tuning (SFT), while pretrained models can effectively separate topics (shown in different colors) in natural language (Figure 3.1a), they struggle with fine-tuning data that deviates from the pretraining distribution (Figure 3.1b). Additionally, the cost of training a reference model escalates with model size (Figure 3.1c), making existing data selection methods for large models prohibitively expensive. 28

3.2 Examples in the same clusters have very similar loss trajectories (Figure 3.2a) while the loss trajectories of examples in different clusters are very different (Figure 3.2b). 34

3.3 Examples in the same clusters of training trajectories on a small model (Pythia-70M) also have similar training trajectories on a large model (Pythia-2.8B), even if the trends may not be the same on both models. 34

3.4 **Data Scaling:** Accuracies (↑) on in-domain and out-of-domain datasets using Pythia-410M. Data size refers to the total number of unique training examples used. All experiments in this table share the same total training steps and learning rate schedule (see Section 3.4.2). See breakdowns in Figure 9.5. 38

3.5 Wall-clock time required to train the reference model and select 100K data from MathInstruct for training Pythia-410M. 38

3.6 Distribution of the coverage of top-1 topic in each cluster. Taller bars on the right end of the plot indicate clusters with a higher concentration of a single topic and therefore suggest a grouping of similar examples. 40

3.7 Performance (↑) of models trained on (1) **random**ly selected 30K examples, (2) **S2L** selected 30K examples, and (3) full 61K examples (**none**) evaluated with 3 different metrics. The minimum value on the y-axis is the performance of the model before fine-tuning. S2L improves the data efficiency for the clinical text summarization task by outperforming training on the full dataset with only less than half of the data. 42

3.8 S2L is robust to the length of training trajectories. 44

3.9 S2L prefers dense trajectories over sparse ones. 44

3.10 Per-dataset and average accuracy comparing proxy training on 100K examples and full data. S2L remains effective. 46

3.11 Per-dataset and average accuracy across different values of the clustering parameter K. S2L is relatively robust to the choice of K. 46

3.12 Relative accuracy to full data across different epochs, comparing S2L-selected data and full data. S2L achieves superior performance with fewer data and fewer training iterations. 46

3.13 Per-dataset and average accuracy comparison between using different proxy models (Pythia-160M and GPT-2 (124M)) for data selection. Using both proxy models show comparable performance, demonstrating the effectiveness of different small models as reference models for S2L. 46

4.1 500 effective (red) and ineffective (purple) poisons crafted by GM and BP in from-scratch and transfer learning scenarios on CIFAR10. (a) Number of effective vs. ineffective poisons and their distance to the target in the (last layer) gradient space of a clean model; (b) Embeddings of effective (red) and ineffective (purple) poisons, and clean examples of the target (blue) and poison (green) class, projected on the first 2 principal components. Effective poisons are not examples with the lowest confidence or highest loss. 54

4.2 Fraction of effective poisons dropped vs fraction of all examples dropped during training on CIFAR10 poisoned with GM, for our method (EPIc) vs lowest-confidence and highest-loss with thresholds .25,.5/1,2 shown by transparent colors, and their average shown in opaque. Left: from scratch. Right: transfer learning. 56

4.3 Training with EPIC on CIFAR10 poisoned with GM. (a) The similarity between effective poisons' gradients to each other becomes small (they get isolated) after the warmup period, (b) EPIC effectively eliminates effective poisons while dropping a smaller fraction of clean examples, (c) EPIC preserves main gradient components, hence remaining examples have a closer gradient to that of the full data, compared to random subsets of the same size. Thus, EPIC preserves the training dynamics. 57

4.4 Fraction of clean vs Gradient Matching poisons in gradient clusters of different sizes, during from-scratch learning with EPIC for 200 epochs. Effective poisons become isolated during training and can be iteratively eliminated by EPIC. . . 59

4.5 Fraction of clean vs Bullseye Polytope poisons in gradient clusters of different sizes, during transfer learning with EPIC for 40 epochs. Effective poisons become isolated during training and can be iteratively eliminated by EPIC. 59

4.6 Attack success rate vs. running time of different defenses, for GM attack on CIFAR-10 with the 40 Epochs pipeline. 68

5.1 Training LeNet-5 on Colored MNIST containing colored handwritten digits. (a) Each digit is a class; the majority of digits in a class have a particular color, and the remaining digits are in 4 other colors. (b) The network output is almost exclusively indicated by the color of the majority group, early in training. (c) Majority and minority groups are separable based on the network output. . . . 76

5.2 GradCAM Visualization. Warmer colors correspond to the pixels that are weighed more in making the final classification. SPARE allows learning the core features instead of spurious ones. 85

5.3 Spurious correlation between"green leaf" & "insect" in Restricted ImageNet found by SPARE. 87

6.1 An overview of the problem, our proposed solution, and the resultant outcomes. (A) We demonstrate the data distribution and provide an example of the statistics of Waterbirds. (B) The overall procedure of PDE. (C) We use GradCAM [162] to show the attention of the model trained with PDE as compared to ERM. While ERM focuses on the background, PDE successfully trains the model to capture the birds. 90

6.2 Visualization of the data. 94

6.3 Comparison of methods in different scenarios. 103

6.4 **Training process of ERM vs. PDE.** We consider the same dataset generated from the distribution as in Definition 6.1.1 for ERM (case 1) and PDE. On the same training data, ERM learns the spurious feature while PDE successfully learns the core feature. We further consider ERM (case 2) when training on the data distribution where $\beta_c > \beta_s$ and $\hat{\alpha} = 0.98$. We show the growth of the max inner product between the model's neuron and core/spurious signal vector and the decrease of training loss with regard to the number of iterations t. 103

6.5 The effect of resetting the momentum after the warm-up stage for PDE on Waterbirds. 106

6.6 The variations in both worst-group and average accuracy on the test set of Waterbirds during the expansion stage under different expansion learning rates. Each vertical dashed line denotes an expansion and the arrow denotes the early stopping. 107

7.1 The baby pacifier class in ImageNet is spuriously correlated with the presence of babies, which leads the pre-trained model to be less accurate for cases when babies are absent in the image (bottom row) and also be right for the wrong reasons when babies are present (top row). Our approach mitigates both concerns by conveniently expressing and decorrelating the spurious relationships in the loss function via language. 111

7.2 Spurious correlation detection based on attributes from an open-vocabulary detector and accuracy discrepancy scores of the model between examples when the spurious attribute is present or absent. 119

7.3 GradCAM explanations for cases when Pre-trained CLIP RN50 relies on the spurious classification described in Table 7.3. 129

7.4 GradCAM explanations for different approaches based on CLIP RN50 for the Waterbirds dataset. 130

7.5 GradCAM explanations for different approaches based on CLIP ViT-L/14@336px for the Waterbirds dataset. 131

7.6 GradCAM explanations for different approaches based on CLIP RN50 for the ImageNet dataset. 132

9.1 Training ResNet-20 on CIFAR-10. (a) Union of mini-batch coresets has a smaller error in capturing the full gradient, compared to the bias of the individual mini-batch coresets. (b) Normalized bias of coresets by the full gradient norm, i.e., $\epsilon = \mathbb{E}[\|\boldsymbol{\xi}_{t_l}\|]/\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l})\|$ in Theorem 2.3.1. Spare coresets have a consistently small $\epsilon < 1$. As the gradient norm becomes smaller closer to the stationary points, small ϵ implies that the bias of the Spare mini-batch coresets $\mathbb{E}[\|\boldsymbol{\xi}_{t_l}\|]$ diminishes closer to the stationary points. Hence, convergence of Spare can be guaranteed (Case 1 in Theorem 2.3.1). On the other hand, ϵ can be large for Craig coresets. Hence, convergence is not guaranteed (Case 2 in Theorem 2.3.1). 146

9.2 Training ResNet-20 on CIFAR-10 with Spare. (a) Dropped examples are learned later in training, by training on Spare subsets. (b) Distribution of forgetting scores for the examples selected by Spare during the training. The distribution is long-tailed, confirming that not all examples contribute equally to training. . 146

9.3 Relative error (%) with 10% training budget. Training on SPARE mini-batch coresets of size $m = 128$ selected from random subsets V_p of size $r = 500$ has a smaller relative error than training on random mini-batches of size $m = 128$. In particular, relative error of SPARE with $m = 128$ is close to that of training on random mini-batches of size $m = 500$. 147

9.4 Variance of gradients of SPARE mini-batches of size $m = 128$ selected from random subsets V_p of size $r = 500$ is very closer to the variance of V_r. In contrast, random subsets of size $m = 128$ have a considerably larger variance. 147

9.5 Break-down accuracies (↑) on in-domain and out-of-domain datasets using Pythia-410M. Data size refers to the total number of unique training examples used. All experiments in this table share the same total training steps and learning rate schedule (see Section 3.4.2). 156

9.6 Examples in the cluster shown in Figure 3.7a: open-formed algebra. Questions are in black and answers are in cyan. 157

9.7 Examples in the cluster shown in Figure 3.7b: reading comprehension + coding. Questions are in black and answers are in cyan; instructions are highlighted in orange. 158

9.8 Examples in the cluster shown in Figure 3.7c: multiple-choice + multi-step reasoning. Questions are in black and answers are in cyan; instructions are highlighted in orange. 159

9.9 Compared to the original topic distribution, SPARE prioritized easier topics (e.g., pre-algebra over intermediate algebra, algebra over other more advanced topics) while always ensuring complete and more balanced coverage of all topics. 160

9.10 **Training process of ERM trained with GD+M.** We consider the same dataset generated in Figure 6.4 and observe almost the same training process as ERM with GD, except GD+M learns the features faster. 185

9.11 **Variation of PDE.** We consider the same dataset generated in Figure 6.4 and add all data at once after the warm-up stage. 185

9.12 GradCAM explanations for different approaches based on CLIP RN50 for the ImageNet dataset, “can opener" class. 203

9.13 The “can opener" class in ImageNet has several concept variations, posing additional challenges in the learning process. The presence of spurious correlations can in addition exacerbate these issues or even hide them when models are right for the wrong reasons (e.g., the last two examples in this figure). While our method reduces the focus on “cans", it is still not able to completely alleviate the problem for examples with unusual concept variation. 204

9.14 Metal features of can openers share visual commonalities with cans, which makes the problem of mitigating spurious correlations more difficult for such cases. . . 205

Acknowledgments

First and foremost, I would like to thank my advisor, Baharan Mirzasoleiman. Over the course of my PhD, I have been deeply inspired by your dedication to research, your incredible attention to detail, and the genuine care you show for your students. Your guidance and mentorship have shaped not only my academic journey but also how I approach challenges in life. As a female researcher and role model, your hard work and resilience have inspired me to aim higher, think deeper, and persevere even in the toughest moments. Being your first student has been an extraordinary privilege. You have been not only a mentor but also my best collaborator, always ready to share ideas, refine experiments, and tackle problems together. Together, we pushed through late nights, perfecting papers and submitting them just in time for deadlines. This dissertation is a reflection of the journey we have taken, and I am truly grateful to have walked this path under your mentorship.

I would also like to thank my committee members, Guy Van den Broeck, Quanquan Gu, and Aditya Grover, for their time, thoughtful questions, and valuable perspectives. Your academic expertise in your respective fields has been a source of inspiration, and I am sincerely grateful for your support.

I am deeply thankful to my mentors and collaborators who have guided and supported me throughout my PhD. During my internships, I had the privilege of working with exceptional teams that significantly shaped my research and professional growth. At Microsoft Research, I would like to thank Besmira Nushi and Hamid Palangi for their guidance and support during a challenging time of remote collaboration amidst COVID. I also fondly recall the family dinner with them and their families during ICML 2023 in Hawaii, which was a memorable moment of connection beyond research. At FAIR at Meta, I am deeply grateful to my mentors Newsha Ardalani, Ari Morcos, and Carole-Jean Wu for their expertise, encouragement, and mentorship, which extended beyond my time at Meta and continues to shape my career. I would also like to thank my collaborators Aaditya K Singh, Mostafa Elhoushi, Anas Mahmoud, Kushal Tirumala, Fabian Gloeckle, Baptiste Rozière, and Hugh Leather for their

valuable contributions and insightful discussions, which made working on these projects truly enjoyable. I also appreciate the time spent in reading groups, as well as the lunches and dinners we shared, where our deep discussions went beyond research. I am thankful for the friendships I built with Aaditya, Kushal, Anas, Amro, William Held, Rylan Schaeffer, and many others, whose talent, hard work, and shared passion for research made our time together truly enriching. These connections have continued beyond our internships, with all of us witnessing and supporting each other through key milestones in our careers.

While wrapping up my dissertation, I joined Virtue AI at its early stage, introduced by my friend Yi Zeng, who knew of my interest in research-driven startups. I am thankful to Bo Li for offering me the opportunity and for her ongoing support. Together, Yi and I developed AI risk taxonomies and safety benchmarks. I was fortunate to collaborate with Percy Liang and Kevin Klyman, whose insights and careful revisions of our manuscripts significantly improved the work. At Virtue AI, I led the code generation research, and I appreciate Wenbo Guo's hands-on approach, as well as the inspiration I gained from Dawn Song's vision. I am also grateful to my mentors Sanmi Koyejo and James Zou, and my collaborator Zinan Lin, for the insightful research discussions. Lastly, I want to thank my peers Yuzhou Nie, Zhun Wang, Chengquan Guo, Chulin Xie, Andy Zhou, and many others, from whom I've learned so much, especially in safety and security, areas where their expertise far exceeds my own.

I am grateful to collaborators from other labs, institutions, and companies who have broadened my perspectives through their unique styles and approaches. Thank you to my mentors Jason Cong, Cho-Jui Hsieh, Kai-Wei Chang, Jeffrey N Chiang, Gintare Karolina Dziugaite, and Zhangyang Wang, as well as peers Neha Prakriya, Hritik Bansal, Fan Yin, Xuxi Chen, and Nishad Singhi.

Within BigML, my research lab at UCLA, I have been fortunate to work alongside incredible lab mates Yihao Xue, Siddharth Joshi, Wenhan Yang, and Dang Nguyen. Each of them has a unique personality, yet they all share a strong dedication to research. Whether through their perspectives, humor, or approaches to their work, there are qualities in each of them that I admire and strive to learn from. I also want to thank the talented interns I have worked with in our lab: Tian Yu Liu (now a PhD student at UCLA), Hao Kang (now a PhD student at Georgia Tech), Eric Gan, Siddhartha Mishra, and Rathul Anand. Collaborating with them has been a rewarding experience.

Before starting my PhD, I had the privilege of working in several labs that laid the foundation for my research journey. At The Center for Vision, Cognition, Learning, and Autonomy (VCLA), I am deeply grateful to Quanshi Zhang and Erik Nijkamp, as well as my advisors Ying Nian Wu and Song-Chun Zhu, for their guidance, which helped me build a strong foundation in research techniques. Later, I had the opportunity to work in Jungseock Joo's Computational Media Lab, where I explored the intersection of computer vision and social science. I am thankful to Jungseock and Seungbae Kim, who supported my research there, as well as to Yue Wu for his mentorship during my internship at Amazon alongside Jungseock, which marked my first research experience in industry. Even after I began my PhD, Jungseock continued to show his generosity by inviting me to a group dinner and writing multiple fellowship recommendation letters for me, for which I am deeply grateful.

I thank the Computer Science Department, especially Joseph Brown, Helen Tran, Juliana Alvarez, and Osanna Kazarian, for their support throughout my PhD journey. I also appreciate the professors and staff of the Mathematics and Statistics Departments, my undergraduate departments, for laying the foundation of my academic path. Finally, UCLA, my true Bruin home, has been a place where I've spent unforgettable years and grown both academically and personally.

To Yihe Deng, who has been my constant through every twist and turn of this journey. She introduced me to the field of machine learning, setting me on the path that led to this PhD, and stood with me as we navigated the challenges of COVID and graduate school together. She has also been my daily partner in research discussions and beyond, always offering fresh perspectives and thoughtful insights. Her courage and persistence have been my counterbalance, challenging me to see beyond my quiet resolve and helping me approach life with greater confidence. Together, we've turned struggles into growth and shared moments into lasting memories. In the City of Angels, we shared late-night drives, endless conversations, and fleeting moments under the city lights that will always stay with me.

Finally, I want to thank my family for their unconditional love and constant support throughout this journey. As the first woman in STEM and the first graduate student in our family, I am deeply grateful for the freedom and encouragement you have always given me. To my parents, thank you for providing me with a childhood free of pressure, where I could explore my passions freely. When I decided to study abroad, you stood behind my choice, even as it meant years of separation. Despite the distance, you have never lacked in offering every form of care—both practical and emotional—through countless video calls and heartfelt conversations, making me feel connected and supported, no matter how far apart we were. To my grandparents, thank you for always celebrating my achievements. To my uncles and aunts, thank you for always spoiling me as your beloved niece. To my younger cousins, thank you for always looking up to me as a role model. I am truly blessed to have such a loving family, who has always been my strongest foundation and safety net, giving me the courage to pursue my dreams.

Curriculum Vitae

2015 – 2019	B.S. in Mathematics of Computation and Statistics, University of California, Los Angeles (UCLA).
2019 – 2021	M.S. in Computer Science, University of California, Los Angeles (UCLA).
2021 – Present	Ph.D. student in Computer Science, University of California, Los Angeles (UCLA).
2021	Applied Scientist Intern, Alexa AI, Amazon.
2022	Research Intern, Microsoft Research.
2023	Research Scientist Intern, FAIR, Meta Platforms.
2024 – Present	Senior Research Scientist, Virtue AI.
2021	Department of Computer Science Fellowship, Graduate Division, UCLA.
2022	Amazon Doctoral Student Fellowship, AWS AI, Amazon.
2024	Dissertation Year Fellowship, UCLA.

Publications

Yu Yang, Tian Yu Liu, and Baharan Mirzasoleiman. "Not all poisons are created equal: Robust training against data poisoning." In Proceedings of the 39th International Conference on Machine Learning, pages 25154–25165. PMLR, 2022.

Tian Yu Liu, Yu Yang, and Baharan Mirzasoleiman. "Friendly noise against adversarial noise: A powerful defense against data poisoning attack." In Advances in Neural Information Processing Systems, 2022.

Yu Yang, Hao Kang, Baharan Mirzasoleiman. “Towards Sustainable Learning: Coresets for Data-efficient Deep Learning.” In Proceedings of the 40th International Conference on Machine Learning, 2023.

Yu Yang, Besmira Nushi, Hamid Palangi, Baharan Mirzasoleiman. “Mitigating Spurious Correlations in Multi-modal Models during Fine-tuning.” In Proceedings of the 40th International Conference on Machine Learning, 2023.

Neha Prakriya, Yu Yang, Baharan Mirzasoleiman, Cho-Jui Hsieh, and Jason Cong. “NeSSA: Near-storage data selection for accelerated machine learning training.” In Proceedings of the 15th ACM Workshop on Hot Topics in Storage and File Systems, pp. 8-15. 2023.

Yihe Deng*, Yu Yang*, Baharan Mirzasoleiman, and Quanquan Gu. “Robust learning with progressive data expansion against spurious correlation.” In Advances in neural information processing systems 36 (2024).

Xuxi Chen*, Yu Yang*, Zhangyang Wang, and Baharan Mirzasoleiman. "Data Distillation Can Be Like Vodka: Distilling More Times For Better Quality." In Proceedings of the Twelfth International Conference on Learning Representations, 2024.

Yu Yang, Eric Gan, Gintare Karolina Dziugaite, Baharan Mirzasoleiman. “Identifying Spurious Biases Early in Training through the Lens of Simplicity Bias.” In Proceedings of The 27th International Conference on Artificial Intelligence and Statistics, 2024.

Yu Yang, Siddhartha Mishra, Jeffrey N Chiang, Baharan Mirzasoleiman. “SmallToLarge (S2L): Scalable Data Selection for Fine-tuning Large Language Models by Summarizing Training Trajectories of Small Models.” In Advances in Neural Information Processing Systems, 2024.

CHAPTER 1

Introduction

The rapid expansion of data availability and the increasing scale of model sizes have revolutionized deep learning, driving transformative breakthroughs in computer vision, natural language processing, and multimodal learning. While larger datasets have enabled substantial performance gains, they also amplify key issues that must be addressed for sustainable and reliable deep learning:

1. **Escalating computational costs**: Training on massive datasets imposes immense demands on computational resources, energy, and time. As datasets and models continue to grow, these costs are becoming increasingly unsustainable, creating barriers to widespread adoption and environmental responsibility.

2. **Susceptibility to adversarial attacks**: Data-driven models are vulnerable to manipulations such as **data poisoning**, where attackers inject malicious examples into the training data. These attacks exploit the model's reliance on patterns within the dataset, compromising performance and security.

3. **Bias and spurious correlations**: Real-world datasets often contain hidden biases or irrelevant correlations. These spurious features lead models to learn shortcuts that degrade their generalization and disproportionately harm underrepresented groups.

To address these challenges, my research adopts a data-centric approach, emphasizing how improved data selection, curriculum design, and weighting strategies can enhance the efficiency and robustness of training deep learning models. Large-scale datasets often contain significant redundancy, adversarial vulnerabilities, and hidden biases, which hinder model performance and scalability. A data-centric perspective tackles these issues by enabling models to focus on the most informative and representative subsets of data, thereby reducing computational demands while preserving performance. Additionally, understanding the relationships and dynamics within the data allows for designing defenses that address vulnerabilities such as adversarial poisoning by identifying problematic patterns in the dataset. Furthermore, addressing spurious correlations within training data ensures that models learn meaningful and generalizable features, promoting fairness and improving reliability in real-world applications.

This approach raises several key research questions:

1. Efficient training requires models to focus on data that provides the most value for learning, avoiding the need to process all available data indiscriminately. How can we systematically identify and prioritize the most informative subsets of data to reduce redundancy and computational costs while maintaining or enhancing model performance?

2. While reducing redundant or low-quality data is beneficial, it cannot address the risks posed by carefully crafted malicious examples. What makes manipulated or poisoned data difficult to detect, and how can we better identify these examples within large datasets?

3. Even when the data is free of manipulation, biases and spurious correlations inherent in the dataset can lead models to learn misleading patterns that degrade fairness and reliability. How can we detect and mitigate these spurious correlations to ensure models focus on meaningful, generalizable features rather than irrelevant or biased ones?

My research aims to answer these questions by proposing effective data selection and training methodologies, which not only address specific challenges in computational efficiency and robustness but also bridge the gap between theoretical insights and practical applications.

1.1 Research Contributions

This dissertation explores challenges in efficient, robust, and fair training of deep learning models, proposing novel data selection and training methodologies to improve data efficiency, robustness against data poisoning, and bias mitigation.

1. **Data Efficiency**

   - **CREST (ICML 2023):** A scalable framework that identifies valuable training examples for non-convex models by modeling loss as piece-wise quadratic and extracting mini-batch coresets. CREST accelerates convergence and provides theoretical guarantees.
   - **S2L (Small-to-Large) (ICLR 2024):** A framework for efficient fine-tuning of large language models by leveraging loss trajectories of smaller proxy models to select informative subsets. S2L reduces computational costs while maintaining performance across tasks like mathematical reasoning and clinical summarization.

2. **Robustness Against Data Poisoning**

   - **EPIC (ICML 2022):** A defense mechanism that identifies and excludes low-density gradient examples to neutralize data poisoning attacks. EPIC maintains generalization performance and offers a lightweight, practical alternative to existing defenses.

3. **Bias Mitigation**

   - **SPARE (ICML 2023):** A method that leverages the simplicity bias of gradient descent to separate and balance spurious correlations early in training. SPARE improves worst-group accuracy across datasets like Waterbirds and CelebA, ensuring fairness in model predictions.
   - **PDE (Progressive Data Expansion) (NeurIPS 2023):** A training pipeline that balances group sizes through staged data expansion, enabling models to

learn progressively complex patterns. PDE achieves state-of-the-art worst-group accuracy and reduces training times by up to 10×.

- **Multimodal Spurious Mitigation (ICML 2023):** A fine-tuning method for multimodal models, such as CLIP, that employs contrastive losses to reduce spurious correlations. This method significantly improves fairness and worst-group accuracy on datasets like Waterbirds and ImageNet.

1.2 Real-world Impacts

The methodologies and frameworks developed in this dissertation have demonstrated tangible real-world benefits:

- **Hardware Efficiency**: Near-storage data selection on SmartSSD (HotStorage 2023) significantly reduced data movement overhead, achieving up to **5.37× speedup** in training tasks by minimizing I/O bottlenecks. CREST's ability to identify and utilize only the most informative training examples enabled this efficiency. By embedding data selection directly into storage hardware, the approach demonstrated scalability for large datasets commonly used in vision tasks, such as ImageNet. This work highlights the potential for integrating machine learning algorithms with emerging hardware solutions to reduce training costs and environmental impact.

- **Medical Applications**: Data-efficient training for clinical text summarization using S2L improved the sustainability of training large language models on the MIMIC-III dataset. S2L demonstrated that a carefully selected **50% subset** of training data, identified via proxy models, could match or even exceed the performance achieved with the full dataset. This method reduces computational costs significantly while maintaining clinical accuracy.

Part I

Data Selection for Efficient Training

CHAPTER 2

CREST: Data-efficient Training for Deep Vision Models

Large datasets have enabled over-parameterized neural networks to achieve unprecedented success [53, 26, 226]. However, training such models, with millions or billions of parameters, on large data requires expensive computational resources, which consume substantial energy, leave a massive amount of carbon footprint, and often soon become obsolete and turn into e-waste [12, 160, 176]. While there has been a persistent effort to improve the performance and reliability of machine learning models [26, 207, 226], their sustainability is often neglected.

Indeed, not all examples are equally valuable or even required to guarantee a good generalization performance. To address the sustainability and efficiency of machine learning, one approach involves selecting the most relevant data for training. A recent line of work [126, 89, 88, 148] showed that for strongly convex models, a weighted subset (coreset) of data that closely matches full gradient—sum of the gradient of all the training examples—provide convergence guarantees for (Incremental) Gradient Descent. Such coresets speed up learning by up to 6x. Intuitively, this is possible as for popular strongly convex models|logistic, linear regression, and regularized support vector machines|the gradient error of a coreset during the *entire training* can be upper-bounded in advanced [126].

Unfortunately, we cannot simply apply the same idea to non-convex models, for three reasons. First, for non-convex models, training dynamics|loss and gradient of examples|drastically change during the training and cannot be upper-bounded beforehand. As a result, the importance of examples for learning changes throughout training, and one coreset cannot guarantee convergence anymore. Second, non-convex models are learned with (mini-batch) stochastic gradient methods, such as (mini-batch) SGD, which require unbiased estimates of the full gradient with a bounded variance. Existing coresets that capture the gradient of the full data cannot provide any guarantee for stochastic gradient methods, as the gradient of mini-batches selected from the coreset may be biased and have a large variance. Finally, iteratively selecting coresets from full data becomes very expensive for large datasets, and does not yield speedup.

In this chapter, we address the above challenges and propose CREST, a rigorous method to find coresets for non-convex models, by making the following contributions:

Coreset selection by modeling the non-convex loss. To ensure a small gradient error throughout training, our key idea is to divide the loss into multiple quadratic sub-regions and find a coreset for learning every quadratic sub-region. To do so, we model the loss of every example as a quadratic function based on its current gradient and curvature information at model parameters $\boldsymbol{w}_{t_l}$. Then, we find a coreset that captures the full gradient at $\boldsymbol{w}_{t_l}$, and keep training on it as long as the quadratic approximated loss of the coreset (sum of quadratic functions corresponding to its elements) closely captures the actual loss of the full data. Otherwise, we update the coreset. In doing so, we ensure a small gradient error during the entire training, which we leverage to guarantee convergence to a stationary point.

Coresets for (mini-batch) stochastic gradient methods. To address coreset selection for (mini-batch) stochastic gradient methods, our idea is to sample multiple subsets of training data uniformly at random, and select a mini-batch coreset from every random sample to closely capture its gradient. The gradients of larger random subsets are unbiased estimates of the full gradient, but have a considerably smaller variance. Hence, the mini-batch coreset gradients are nearly unbiased, and have a small variance. Updating the mini-batch coresets based on above piece-wise quadratic loss approximation, ensures a small bias throughout training. This allows providing superior convergence guarantee for training with stochastic gradient methods. Besides, it significantly improves the computational complexity of finding coresets and scales coreset selection to much larger datasets.

Further improving the efficiency of coreset selection. To further improve the efficiency and scalability of coreset selection, we make the following observation. When a group of examples are learned, their gradients become nearly zero. Hence, a few examples can well represent the gradient of the corresponding group and the entire group can be safely excluded from coreset selection afterwards. CREST iteratively excludes examples that are learned and have a very small loss during multiple consecutive training iterations, and finds mini-batch coresets from the remaining examples. This speeds up learning, and improves the efficiency and performance of coreset selection, in later stages of training.

Through extensive experiments, we demonstrate the effectiveness of CREST for training various over-parameterized models on different vision and NLP benchmark datasets, including ResNet20 on CIFAR-10 [9], ResNet18 on CIFAR-100 [9], ResNet-50 on TinyImageNet [154], and RoBERTa [108] on SNLI [23] with 570K examples. CREST is able to achieve 1.7x to 2.5x speedup over training on the full data, while introducing the smallest relative error compared to the baselines. To our knowledge, this is the first time coreset selection has been applied to such large models and datasets in vision and NLP.

Finally, we analyze the examples selected by CREST at different times during the training. We quantify the learning difficulty of every example using the forgettability score [183], which counts the number of times an example is misclassified after being correctly classified during training. We find that early in training, the most effective subsets for learning deep models are easy-to-learn examples. As training proceeds, the model learns the most from examples with increasing levels of learning difficulty. Interestingly, the model never requires training on easiest-to-learn examples to achieve a good generalization performance.

2.1 Related Work

Several heuristics have been recently proposed for finding coresets for training machine learning models. A line of work first fully trains the original model [19] or a smaller proxy [42]. Then, it selects examples with the most centrally located embeddings [19], highest uncertainty, i.e., the entropy of predicted class probabilities [42], largest forgetting events, i.e., the number of times an example is misclassified after being correctly classified [183], or large expected gradient norm over multiple initializations [143]. These methods do not yield any speedup or theoretical guarantees.

Another line of work selects examples during training to speed up learning. Importance sampling techniques employ the gradient norm [8, 87] or the loss [111, 159] to reduce the variance of stochastic gradients during the training. However, importance sampling does not provide rigorous convergence guarantees and cannot provide a notable speedup for training deep models. [122] finds examples that are non-noisy, non-redundant, task-relevant, and reduce the loss on a holdout set the most. This method speeds up training but requires a validation set and does not guarantee convergence.

Most relevant to our work are recent theoretically rigorous techniques that select coresets by iteratively matching the (preconditioned) gradient of full training data, namely [126, 88, 148], or validation set [89]. Such methods guarantee convergence to a near-optimal solution, for training (strongly) convex or nearly convex models under Polyak-Lojasiewicz (PL) condition, using Incremental Gradient (IG) methods, or Gradient Descent (GD). However, they do not guarantee convergence for training non-convex models trained with (mini-batch) stochastic gradient methods, and do not scale to very large datasets. Our work addresses the above shortcomings by developing a rigorous and scalable framework to extract coresets for data-efficient deep learning.

2.2 Problem Formulation and Background

The standard approach to training machine learning models is empirical risk minimization (ERM). Formally, given a loss function $\mathcal{L}$, we find the model parameters $\boldsymbol{w}$ that minimize the expected loss on training examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ indexed by $V=\{1,\cdots,n\}$, sampled from distribution $\mathcal{D}$:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}\in\boldsymbol{W}} \mathcal{L}(\boldsymbol{w}) := \mathbb{E}_{(\boldsymbol{x}_i, y_i)\sim\mathcal{D}}[\mathcal{L}(\boldsymbol{w}; (\boldsymbol{x}_i, y_i))]. \tag{2.1}$$

For over-parameterized models trained on large training data, GD becomes prohibitively slow. Hence, stochastic gradient methods, such as mini-batch SGD are employed in practice. Such methods select one or a mini-batch $\mathcal{M}$ of m examples sampled i.i.d. from the training data, and iteratively step in the negative direction of the stochastic gradient of the sampled examples, scaled by step-size η:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \frac{1}{m} \sum_{i \in \mathcal{M}} \mathbf{g}_{t,i}, \tag{2.2}$$

where $\mathbf{g}_{t,i} = \nabla \mathcal{L}_i(\boldsymbol{w}_t) := \nabla \mathcal{L}(\boldsymbol{w}_t; (\boldsymbol{x}_i, y_i))$ is the gradient of example i at iteration t. Random examples have an unbiased gradient with a bounded variance, i.e., $\mathbb{E}_{i \in V}[\|\mathbf{g}_{t,i} - \nabla \mathcal{L}(\boldsymbol{w}_t)\|^2] \leq \sigma^2$. Hence, they guarantee convergence with an $\mathcal{O}(1/\sqrt{t})$ rate to a stationary point of a non-convex loss [61]. Importantly, random mini-batches of size m have an unbiased gradient with a reduced variance of σ^2/m. As long as mini-batches are not too large, mini-batch SGD achieves a faster convergence rate of $\mathcal{O}(1/\sqrt{mt})$ [197, 82].

Existing coreset methods, such as Craig [126], GradMatch [88], and AdaCore [148] find weighted subsets of examples that match the full training gradient (preconditioned on Hessian). Formally, the goal is to find the smallest subset $S \subseteq V$ and corresponding per-element step-sizes (weights) $\gamma_j > 0$ that approximate the full gradient with an error at most $\epsilon > 0$ for all the possible values of $\boldsymbol{w}_t \in \boldsymbol{W}$:

$$S^* = \underset{S \subseteq V, \gamma_j \geq 0 \ \forall j \in S}{\arg\min} |S|, \ \text{ s.t. } \max_{\boldsymbol{w}_t \in \boldsymbol{W}} \| \sum_{i \in V} \mathbf{g}_{t,i} - \sum_{j \in S} \gamma_j \mathbf{g}_{t,j} \| \leq \epsilon. \tag{2.3}$$

Problem (2.3) requires calculating the maximum gradient error between full and coreset gradient for all $\boldsymbol{w}_t \in \boldsymbol{W}$, which cannot be computed. To address this, [126] showed that for several classes of (strongly) convex problems, including regularized linear and ridge regression, and support vector machines (SVMs), the normed gradient difference between data points *during the entire training* can be efficiently upper-bounded by the difference between feature vectors. This allows turning Problem (2.3) into the following submodular[1] cover problem:

$$S^* = \arg\min |S| \quad \text{s.t.} \quad C - \sum_{i \in V} \min_{j \in S} \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq C - \epsilon, \tag{2.4}$$

[1] A set function $F : 2^V \to \mathbb{R}^+$ is submodular if $F(S \cup \{e\}) - F(S) \geq F(T \cup \{e\}) - F(T)$, for any $S \subseteq T \subseteq V$ and $e \in V \setminus T$. F is *monotone* if $F(e|S) \geq 0$ for any $e \in V \setminus S$ and $S \subseteq V$.

where C is a big constant. A near-optimal coreset of size k can be found from a ground-set of n elements, using the greedy algorithm with complexity of $\mathcal{O}(n \cdot k)$ as a pre-processing step before training. The weights γ_j are calculated as the number of examples $i \in V$ for which $j \in S$ minimizes $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$. This approach has been adopted by [89, 148]. [88] used orthogonal matching pursuit (OMP) to directly find a weighted coreset, by minimizing the regularized objective in RHS of Problem (2.3). However, OMP provides weaker guarantees than greedy, and does not always find a large enough subset. Hence, the coreset needs to be augmented with random examples.

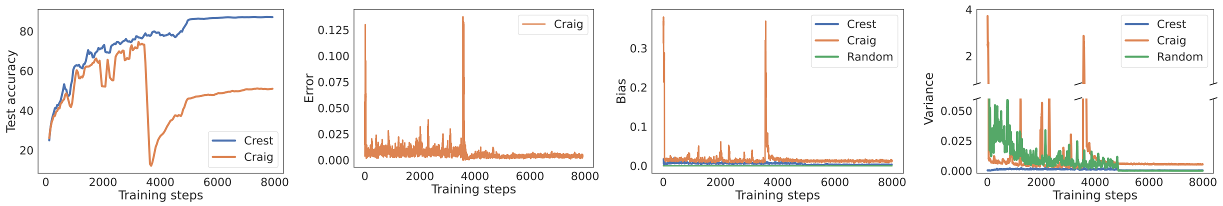
For neural networks, finding coresets based on their very high-dimensional gradients is slow and does not yield high-quality coresets. Instead, one can use the gradient of the loss w.r.t the input to the last layer that is shown to capture the variation of the gradient norm well [87]. Such lower-dimensional gradients $\mathbf{g}_{t,i}^L$ can be quickly obtained with a forward pass and can be used instead of the full gradient to find coresets during the training [124, 148, 89]. Moreover, with a fixed training budget one can find a subset of size k by solving the following submodular maximization problems, which is the dual of the submodular cover Problem (2.4):

$$S_t^* = \arg\max_{S \subseteq V} C - \sum_{i \in V} \min_{j \in S} \|\mathbf{g}_{t,i}^L - \mathbf{g}_{t,j}^L\|, \text{ s.t. } |S| \leq k. \tag{2.5}$$

However, it is not clear when the coresets should be updated to guarantee convergence for training non-convex models. Besides, finding coresets from the full data does not scale to training on large datasets. Importantly, the above method only guarantees convergence for (Incremental) GD and cannot guarantee convergence for stochastic gradient methods used for training neural networks, as we will discuss next.

2.3 Coresets for Training Non-convex Models

In this section, we will first discuss the challenges of extracting coresets for deep models, and then introduce our proposed method, Crest, to overcome the above challenges and make coreset selection applicable to neural networks.



(a) Test Accuracy (b) Error of coreset gradients (c) Bias of mini-batch grads. (d) Variance of mini-batch grads.

Figure 2.1: Training ResNet20 on CIFAR-10. (a) 10% Craig coresets selected at the beginning of every epoch from full data may perform very poorly. This is because, **(C1)**: (b) Coresets may have a large error: $\|\mathbf{g}_{t,S} - \nabla\mathcal{L}(\boldsymbol{w}_t)\|$, after a few training iterations; and **(C2)**: Gradient of weighted mini-batches selected from the coresets may have a (c) large bias $\|\mathbb{E}_i[\mathbf{g}_{t,\mathcal{M}_i}] - \nabla\mathcal{L}(\boldsymbol{w}_t)\|$ and (d) large variance $\mathbb{E}_i[\|\mathbf{g}_{t,\mathcal{M}_i} - \nabla\mathcal{L}(\boldsymbol{w}_t)\|^2]$, where $\mathcal{M}_i \in S$ is a mini-batch and $\mathbf{g}_{t,\mathcal{M}_i} = \mathbb{E}_{j\in\mathcal{M}_i}[\gamma_j \mathbf{g}_{t,j}]$. In contrast, our Crest coresets are nearly unbiased, and have a smaller variance than random mini-batches of same size.

Challenges. The non-convex loss landscape of the non-convex models makes coreset selection very challenging. Fig. 2.1a shows that existing coreset selection methods such as Craig [126] that find coresets by iteratively solving Eq. (2.5) at every epoch may perform very poorly for training deep networks, for the following reasons:

(C1) For deep networks, the loss functions associated with different data points $\mathcal{L}_i$ change very rapidly [47]. Therefore, in contrast to (strongly) convex functions, for which the gradient error of a coreset throughout training can be effectively upper-bounded in advance, e.g., using their feature vectors, such upper bounds cannot be computed for neural networks. That is, even within a relatively small neighborhood $\mathcal{N}$ around $\boldsymbol{w}_t$, the gradient $\nabla\mathcal{L}_i(\boldsymbol{w}_t)$ may be drastically different than $\nabla\mathcal{L}_i(\boldsymbol{w}_t + \boldsymbol{\delta})$ for $\boldsymbol{w}_t + \boldsymbol{\delta} \in \mathcal{N}$. Figure 2.1b shows that the gradient error of coresets found by Craig can be very large after a few training iterations. Here, the challenge is to compute the size of the neighborhood in which a coreset closely captures the full gradient, and update the coreset otherwise.

(C2) Coresets found from the full training data guarantee convergence for (Incremental) GD, but cannot provide any guarantee for *stochastic* gradient methods, such as (mini-batch) SGD, that are applied to train neural networks. This is because stochastic methods require unbiased estimates of the full gradient with a bounded variance. However, as the error of the coresets found from the full data may increase during the training, the gradient of mini-batches selected from the coresets may have a large bias. Besides, as some examples may have a very large weight, the variance of weighted mini-batch gradients are much larger than the variance of random (unweighted) mini-batch gradients used when training on the full data. Figure 2.1c, 2.1d show that gradient of mini-batches selected from the coreset can have a very large bias and variance. Here, the challenge is to find coresets that are nearly unbiased and have a small variance.

(C3) For deep networks, the importance of examples for learning changes over time and hence the coresets should be updated frequently. The greedy algorithm has a complexity of $\mathcal{O}(n \cdot k)$ to find k out of n examples. For large datasets, this prevents the coreset selection methods to achieve a significant speedup. Hence, the challenge is to improve the efficiency of coreset selection for training deep networks.

Next, we discuss how we overcome the above challenges.

2.3.1 Modeling the Non-convex Loss Function

To address the challenge **(C1)** of finding the size of the neighborhood in which a coreset closely captures the full gradient, we model the non-convex loss as a piece-wise quadratic function. In doing so, we reduce the problem of finding coresets for a non-convex objective to finding coresets for a series of quadratic problems. Formally, at every coreset selection step l, we find a coreset S_l that captures the full gradient at $\boldsymbol{w}_{t_l}$. Then, we make a quadratic loss approximation $\mathcal{F}^l$ based on the gradient and curvature of the coreset at $\boldsymbol{w}_{t_l}$. We keep training on the coreset S_l within the neighborhood $\mathcal{N}_l$ in which the quadratic approximation closely follows the actual training loss, i.e., we have that $\mathcal{L}(\boldsymbol{w}_{t_l} + \boldsymbol{\delta}) = \mathcal{F}^l(\boldsymbol{\delta}) \ \forall \ \boldsymbol{w}_{t_l} + \boldsymbol{\delta} \in \mathcal{N}_l$. Otherwise, we update the coreset and make a new quadratic approximation. This ensures a small gradient error within $\mathcal{N}_l$, and similarly through the entire training. Hence, convergence can be guaranteed.

In this work, we extract the coresets by greedily solving the submodular Problem (2.5). But, our piece-wise quadratic approximation can be generally applied to any coreset selection method to check the validity of the coresets, and updating them to guarantee convergence for deep learning.

In the rest of this section, we first discuss how to efficiently estimate the coreset loss as a quadratic function $\mathcal{F}^l$, based on its gradient and curvature at $\boldsymbol{w}_{t_l}$. Then, we discuss finding the size of the neighborhood $\mathcal{N}_l$ in which the quadratic function $\mathcal{F}^l$ closely captures the loss $\mathcal{L}$ of full training data.

Approximating coreset losses by quadratic functions. We model the coreset loss within the neighborhood $\mathcal{N}_l$ by a quadratic approximation $\mathcal{F}^l$, using the 2$^{\text{nd}}$-order Taylor series of expansion of $\mathcal{L}(\boldsymbol{w}_{t_l})$ at $\boldsymbol{w}_{t_l}$ where the coreset is extracted:

$$\mathcal{F}^l(\boldsymbol{\delta}) = \frac{1}{2}\boldsymbol{\delta}^T \mathbf{H}_{t_l,S_l}\boldsymbol{\delta} + \mathbf{g}_{t_l,S_l}\boldsymbol{\delta} + \mathcal{L}(\boldsymbol{w}_{t_l}), \tag{2.6}$$

where $\mathbf{H}_{t_l,S_l}{=}\frac{1}{|S_l|}\sum_{j\in S_l}\gamma_j \mathbf{H}_{t_l,j}$ and $\mathbf{g}_{t_l,S_l}{=}\frac{1}{|S_l|}\sum_{j\in S_l}\gamma_j \mathbf{g}_{t_l,j}$ are the weighted mean of the Hessian and gradient of the examples in the coreset S_l. Such modeling is the main idea behind the popular convexification technique in mathematical optimization, which powers Levenberg-Marquardt [119] and K-FAC [120] optimization methods, among others [15, 31, 197].

To obtain an efficient estimate of the Hessian of the coreset, we use an approximate Hessian operator instead of the full Hessian. Specifically, we employ the Hutchinson's trace estimator method [75] to obtain a stochastic estimate of the coreset Hessian diagonal [20, 49, 209, 219], without having to form the Hessian matrix explicitly:

$$\text{diag}(\mathbf{H}_{t_l,S_l}) = \mathbb{E}[\boldsymbol{z} \odot (\mathbf{H}_{t_l,S_l}\boldsymbol{z})]. \tag{2.7}$$

This method approximates the Hessian diagonal as the expectation of Hessian $\mathbf{H}_{t_l,S_l}$ multiplied by a random vector $\boldsymbol{z}$ with Rademacher distribution. The multiplication $\mathbf{H}_{t_l,S_l}\boldsymbol{z}$ can be efficiently calculated via backprop on gradients of the coreset multiplied by $\boldsymbol{z}$. i.e., $\mathbf{H}_{t_l,S_l}\boldsymbol{z} = \partial \mathbf{g}_{t_l,S_l}^T \boldsymbol{z} / \partial \boldsymbol{w}_{t_l}$.

As the local gradient and curvature information can be very noisy for neural networks [220], to better approximate the global gradient and Hessian information, we smooth them out by applying exponential averaging with parameters $0 < \beta_1$, $0 < \beta_2 < 1$:

$$\overline{\mathbf{g}}_{t_l,j} = \frac{(1-\beta_1)\sum_{t=1}^{t_l}\beta_1^{t_l-t}\mathbf{g}_{t,j}}{1-\beta_1^{t_l}}, \tag{2.8}$$

$$\overline{\mathbf{H}}_{j,t_l} = \sqrt{\frac{(1-\beta_2)\sum_{t=1}^{t_l}\beta_2^{t_l-t}\mathrm{diag}(\mathbf{H}_{t,j})\mathrm{diag}(\mathbf{H}_{t,j})}{1-\beta_2^{t_l}}}. \tag{2.9}$$

For very large networks, gradient and Hessian diagonal w.r.t. the input to the penultimate layer can be used in Eq. (2.7-2.9).

Estimating the size of the quadratic neighborhoods. To check the validity of the coreset, we iteratively compare the value of the quadratic loss $\mathcal{F}^l(\boldsymbol{\delta})$ with the value of the actual training loss. For efficiency, we obtain an unbiased estimate of the actual loss on a small random sample of training examples $V_r \subseteq V$, i.e., $\mathcal{L}^r$. We update the coreset S_l and the quadratic approximation $\mathcal{F}^l(\boldsymbol{\delta})$, when the quadratic coreset loss does not closely follow the actual loss estimate $\mathcal{L}^r(\boldsymbol{\delta}+\boldsymbol{w}_{t_l})$. More precisely, every T_1 iterations, we compute the ratio of the absolute loss difference to the actual loss, i.e.,

$$\rho_{t_l} = \frac{|\mathcal{F}^l(\boldsymbol{\delta}) - \mathcal{L}^r(\boldsymbol{\delta}+\boldsymbol{w}_{t_l})|}{\mathcal{L}^r(\boldsymbol{\delta}+\boldsymbol{w}_{t_l})}. \tag{2.10}$$

We consider the quadratic approximation of the coreset loss to be sufficiently accurate if ρ_{t_l} is smaller than a threshold τ. If $\rho_{t_l} \le \tau$, we keep using the coreset for the T_1 subsequent iterations. Otherwise, we find a new coreset and update the quadratic approximation, accordingly. Computing ρ_{t_l} can be done quite efficiently. $\mathcal{F}^l(\boldsymbol{\delta})$ can be efficiently calculated based on the gradient and Hessian of the coreset using Eq. (2.7-2.9). $\boldsymbol{\delta}$ is the total amount of updates calculated by the optimization algorithm in T_1 training iterations. Calculating $\mathcal{L}^r(\boldsymbol{\delta}+\boldsymbol{w}_{t_l})$ requires an additional forward pass on a subset V_r of data, which we only need once every T_1 iterations.

Remark. In the initial phase of training, gradients evolve very rapidly. Hence, early in training, the quadratic approximations are accurate in a small neighborhood $\mathcal{N}_l$. Therefore, it is crucial to update the coresets frequently to be able to closely capture the full gradient. In contrast, in the final stage of training, the loss becomes well approximated as a convex quadratic within a sufficiently large neighborhood of the local optimum [120]. Hence, the same subset can be used for several training iterations. We show in our experiments that for a fixed τ, CREST updates the coresets much less frequently as training proceeds. In practice, T_1 can grow proportional to the inverse of the norm of the Hessian diagonal, as we confirm experimentally.

2.3.2 Coresets for (Mini-batch) Stochastic GD

Next, we address the challenge **(C2)** of finding coresets for (mini-batch) stochastic gradient methods, that are used for training deep networks. To address this problem, our main idea is to sample multiple subsets of training data $\{V_1, \cdots, V_P\}$ uniformly at random, and directly select a smaller coreset $S_l^p, p \in [P]$ of the mini-batch size m from each random subset V_p. Effectively, instead of selecting a subset to capture the full gradient at $\boldsymbol{w}_{t_l}$, we select multiple *mini-batch coresets* $\{S_l^1, \cdots, S_l^P\}$ at $\boldsymbol{w}_{t_l}$, where each coreset S_l^p is of mini-batch size m, and captures the full gradient of a random subset V_p of training data at $\boldsymbol{w}_{t_l}$.

Formally, at every coreset selection iteration l, we solve P smaller submodular maximization problems. I.e. for $p \in [P]$:

$$S_l^{p*} = \underset{S \subseteq V_p}{\arg\max}\ C - \sum_{i \in V_p} \min_{j \in S} \|\mathbf{g}_{t_l,i}^L - \mathbf{g}_{t_l,j}^L\|, \ \text{ s.t. } |S| \leq m, \tag{2.11}$$

where $\mathbf{g}_{t_l,i}^L$ is the gradient of the loss w.r.t. the input to the last layer of the network at $\boldsymbol{w}_{t_l}$.

Then, we make a quadratic loss approximation of the form Eq. (2.6) to the *union* of mini-batch coresets $S_l = \bigcup_{p \in [P]} S_l^p$. Each random subset V_p provides an unbiased estimate of the full gradient, and since each mini-batch coreset S_l^p closely captures the gradient of V_p, it provides a nearly unbiased estimate of the full gradient. Therefore, the union of the mini-batch coresets S_l also captures the full gradient. However, S_l has a smaller error in capturing the full gradient compared to each of the mini-batch coresets, as small errors of mini-batch coresets cancel each other out (*c.f.* Figure 9.1a in Appendix 9.1.2). Hence, the union of mini-batch coresets makes a more accurate approximation to the full loss.

As long as the quadratic approximation is valid, we can train on *any* of the mini-batch coresets found at w_{t_l}. Thus, we keep selecting mini-batch coresets at random from $\{S_l^1, \cdots, S_l^P\}$, and training on them, as long as the quadratic approximation on the union of selected mini-batch coresets accurately captures the full loss according to Eq. (2.10).

Notably, mini-batch coresets selected from random subsets are nearly unbiased and have a very small variance (*c.f.* Figure 2.1c, 2.1d). This is because random subsets V_p of size r are unbiased and have a r/m times smaller variance than that of random mini-batches of size m. As long as random subsets are not too large, mini-batch coresets capture the gradient of random subsets very closely. This ensures that the gradients of mini-batch coresets are nearly unbiased and have a nearly r/m times smaller variance than random mini-batches of same size (*c.f.* Figure 9.4 in Appendix). Note that there is a trade-off. For a fixed mini-batch size, selecting mini-batch coresets from larger random subsets results in a smaller variance but may introduce a larger bias. The very small bias of CREST mini-batch coresets allows guaranteed convergence to a stationary point. At the same time, their smaller variance ensures superior convergence rate compared to training on full data, as we will show in Theorem 2.3.1. This cannot be achieved by coresets capturing the full gradient.

Note that selecting mini-batches from smaller random partitions speeds up the coreset selection, by breaking one large problem into smaller ones. For example, using the greedy algorithm to solve the submodular maximization Problem (2.5) has a complexity of $\mathcal{O}(n.k)$ to select k examples from a ground-set of n examples. But, solving Eq. (2.11) to select P mini-batches of size k/P from random subsets of size r has a total complexity of $\mathcal{O}(P \times r \cdot \frac{k}{P}) = \mathcal{O}(r \cdot k)$.

Remark. Early in training, quadratic approximations are accurate in a small neighborhood $\mathcal{N}_l$. Hence, a smaller number of mini-batches can be extracted simultaneously. In the final stage of training, the loss can be well captured by a quadratic function [120]. Hence, a larger number of mini-batches can be selected simultaneously later in training. In practice,

Algorithm 1 CoREsets for STochastic GD (CREST)

```
Require: Model parameter $\boldsymbol{w}_0$, mini-batch size $m$, random partition size $r$, learning rate $\eta$,
  total training iterations $N$, checking interval $T_2$, multipliers $b, h$, thresholds $\alpha, \tau$.
  $t \leftarrow 0$, $T_1 \leftarrow 1$, update $\leftarrow 1$
  while $t < N$ do
    if update $== 1$ then
      for $p = 1$ to $P$ do
        Select a random subset $V_p \subseteq V$ s.t. $|V_p| = r$
        $S_l^p \in \arg\max_{\substack{S \subseteq V_p \\ |S| \leq m}} C - \sum_{i \in V_p} \min_{j \in S} \|\mathbf{g}^L_{t_l,i} - \mathbf{g}^L_{t_l,j}\|$
      end for
      $S_l = \bigcup_{p \in [P]} S_l^p$
      Calculate $\mathcal{F}^l$ with $\overline{\mathbf{H}}_{t,S_l}, \overline{\mathbf{g}}_{t,S_l}$
    end if
    for $j = 1$ to $T_1$ do
      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \mathbf{g}_{S_l,t}$
      $t \leftarrow t + 1$
      if $t \mod T_2 == 0$ then
        $V = \{j \in V | \mathcal{L}_j(\boldsymbol{w}_i) > \alpha, \forall i \in [t - T_2, t]\}$.
      end if
    end for
    $\boldsymbol{\delta} \leftarrow \boldsymbol{w}_t - \boldsymbol{w}_{t-T_1}$
    Calculate $\rho_t$ from Equation (2.10).
    if $\rho_t > \tau$ then
      update $\leftarrow 1$,
      $T_1 \leftarrow h \times \|\overline{\mathbf{H}}_0\| / \|\overline{\mathbf{H}}_t\|$,  $P \leftarrow b \times T_1$
    else
      update $\leftarrow 0$
    end if
  end while
```

simply increasing P proportional to the inverse of the norm of the Hessian diagonal works well, as we confirm by our experiments.

2.3.3 Further Improving Efficiency of Coreset Selection

To address the challenge **(C3)** of further improving the efficiency and performance of selecting coresets, we make the following observation. Examples are gradually learned during the training. When an example is learned, its gradient and loss become nearly zero. Hence, such examples do not affect training and can be dropped from the coreset selection pipeline to improve efficiently. However, the gradient or loss of an example at a single point during training can be very noisy. To quickly identify such examples, we monitor the loss of examples within non-overlapping intervals of length T_2 during the training, and exclude those that consistently have a loss smaller than a threshold α. This shrinks the size of the selection problem over time, and allows CREST to focus more on examples that are not learned. Hence, it further improves the efficiency and speedup of the algorithm.

To efficiently exclude the learned examples, we only rely on the loss values calculated for random subsets used for selecting the coresets, and drop examples for which the calculated loss values are smaller than α in an interval of length T_2.

Effectively, dropping the learned examples speeds up training by increasing the learning rate. Specifically, dropping s examples with nearly-zero gradients from a ground-set of n examples increases the full (average) gradient by nearly $n/(n-s)$, which has a similar effect to that of increasing the learning rate by a factor of nearly $n/(n-s)$.

The pseudocode of CREST is illustrated in Alg. 1.

The following Theorem shows that training with stochastic gradient descent on mini-batch coresets found by CREST converges to a stationary point of the non-convex loss.

Theorem 2.3.1. *For any $\delta, \lambda > 0$, assume that the function $\mathcal{L}$ is L-gradient Lipschitz, and stochastic gradients $\mathbf{g}_{t,i}$ have a bounded variance, i.e., $\mathbb{E}_{i\in V}[\|\mathbf{g}_{t,i} - \nabla\mathcal{L}(\boldsymbol{w}_t)\|^2] \leq \sigma^2$.*

Case 1 (CREST: Nearly-unbiased). *Let step size be $\eta = \min\{\frac{1}{L}, \frac{\tilde{D}\sqrt{r}}{\sigma\sqrt{N}}\}$, for some $\tilde{D}$>0 and N be the number of training iterations. If the gradient bias of mini-batch coresets $\mathbb{E}[\|\boldsymbol{\xi}_{t_l}\|] \leq \epsilon\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l})\|$ and $\tau \leq \min_l(\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l} + \boldsymbol{\delta}_l)\| - 3\epsilon\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l})\|)\|\boldsymbol{\delta}_l\|/2\mathcal{L}(\boldsymbol{w}_{t_l} + \boldsymbol{\delta}_l)$ for $0 \leq \epsilon \leq \min\{1, \|\nabla\mathcal{L}(\boldsymbol{w}_{t_l} + \boldsymbol{\delta}_l)\|/3\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l})\|\}$, then with probability at least $1-\lambda$, CREST will visit a ν-stationary point at least once in the following number of iterations:*

$$\tilde{\mathcal{O}}\left(\frac{L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{\nu^2}(1 + \frac{\sigma^2}{r\nu^2})\right). \tag{2.12}$$

Case 2 (Biased). *If the bias of mini-batches $\mathbb{E}[\|\boldsymbol{\xi}_t\|] \leq \epsilon$, but ϵ is larger than the full* *gradient norm anytime during the training, then the number of iterations is:*

$$\tilde{\mathcal{O}}\left(\frac{L(\mathcal{L}(\boldsymbol{w}_0)-\mathcal{L}^*)}{\nu^2-\epsilon}(1+\frac{\sigma^2+r\epsilon^2}{r(\nu^2-\epsilon)})\right). \tag{2.13}$$

In particular, if $\epsilon \geq \nu^2$, convergence is not guaranteed.

Table 2.1: Relative error (%) of different methods over training on the full data. All the baselines select subsets of size 10% of full data at the beginning of every epoch. On the other hand, Crest selects mini-batches and decides when to update the mini-batches based on its quadratic loss approximation. (*) Glister uses the validation set, and (‡) GradMatch uses higher dimensional gradients to find coresets. SGD† shows accuracy of a standard mini-batch SGD pipeline at 10% training.

Dataset - Model	Backprop	SGD†	Random	Craig	Grad-match‡	Glister*	**Crest** (Ours)
CIFAR-10 - ResNet-20	10%	$21.3_{\pm 8.0}$	$7.2_{\pm 1.4}$	$13.0_{\pm 5.1}$	$6.0_{\pm 0.1}$	$7.0_{\pm 0.1}$	$\mathbf{5.5_{\pm 0.2}}$
CIFAR-100 - ResNet-18	10%	$36.5_{\pm 2.9}$	$11.7_{\pm 0.4}$	$17.2_{\pm 4.5}$	$12.7_{\pm 0.9}$	$27.6_{\pm 4.0}$	$\mathbf{9.4_{\pm 0.3}}$
TinyImageNet - ResNet-50	10%	$32.8_{\pm 2.1}$	$16.0_{\pm 0.5}$	$28.5_{\pm 0.6}$	$27.7_{\pm 0.2}$	$32.8_{\pm 2.1}$	$\mathbf{15.4_{\pm 0.6}}$
SNLI - RoBERTa (Finetune)	10%	$1.2_{\pm 0.3}$	$1.2_{\pm 0.3}$	-	-	-	$\mathbf{0.8_{\pm 0.2}}$

The proof can be found in Appendix 9.1.1. At a high level, Theorem 2.3.1 shows that if mini-batch coresets closely capture gradient of random subsets V_p, Crest with a small enough τ, converges to a ν-stationary point of the non-convex loss, but r/m times faster than mini-batch SGD with mini-batch size m on full data, as discussed next.

Case 1. As Crest mini-batch coresets capture the gradient of random subsets closely, the bias of mini-batch coresets is a small fraction, $\epsilon \in [0,1]$, of the full gradient norm at selection time. If $\epsilon \leq \min\{1, \|\nabla\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)\|/3\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l})\|\}$, a small enough τ ensures that the bias stays smaller than the full gradient norm within the neighborhood $\mathcal{N}_l$ (*c.f.* Figure 9.1b in Appendix 9.1.2). Importantly, as the gradient norm shrinks close to a stationary point, a small ϵ implies that the bias in the entire neighborhood vanishes close to convergence. This guarantees convergence of Crest to a ν-stationary point. Notably, as long as $r \leq \sigma^2/\nu^2$, training with Crest linearly speeds up training by a factor of r. In particular, compared to SGD with mini-batch size m, Crest speeds up training by a factor of r/m.

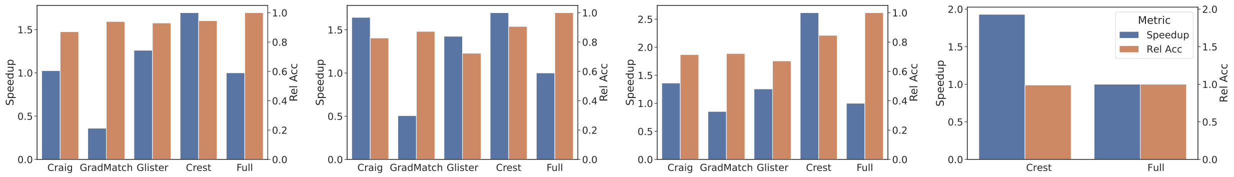
Case 2. If the bias of the mini-batch gradients ϵ is is larger than the full gradient norm or larger than ν, (mini-batch) SGD does not converge to a ν-stationary point. This explains why larger bias of mini-batches selected from coresets extracted from the full data results in a poor performance (*c.f.* Figure 2.1a). Besides, the ϵ bias slows down the training by a factor of $\nu^2 - \epsilon$. Note that such mini-batches also have a larger variance than mini-batch coresets found by CREST, which should be replaced by $1/r$ in Eq. (2.13).

2.4 Experiments

In this section, we evaluate the performance of our coreset selection, CREST. First, we compare CREST to the state-of-the-art coreset selection algorithms, namely CRAIG [126], GLISTER [89], and GRADMATCH [88], as well as the Random baseline. Second, we evaluate the effectiveness of our quadratic approximations in determining the time that the coresets needs to be updated. In addition, we compare the speedup of training with CREST to other baselines. Then, we conduct an ablation study to investigate the necessity of the quadratic vs. linear approximation, smoothing gradient and curvature, and dropping the learned examples from the selection pipeline. Finally, we study the learning difficulty of subsets that are selected by CREST during the course of training.

Datasets and Models. To demonstrate the effectiveness of CREST across different datasets and architectures, we apply CREST to several image and language benchmarks, including training ResNet-20 on CIFAR10, ResNet-18 on CIFAR-100 [9], ResNet-50 on TinyImageNet [154], and fine-tuning RoBERTa on Stanford Natural Language Inference (SNLI) [23]. Table 9.1 summarizes the datasets and models.

Training Setup. For all datasets except SNLI, we consider a standard deep learning training pipeline that runs for 200 epochs with a SGD optimizer with a momentum of 0.9, and decays the learning rate by a factor of 0.1 after 60% and 85% of training, and use mini-batch size 128. We warm-start the learning rate to 0.1 in the first 10% of training, which is essential for stability of all the methods, except CREST and Random. However, for fair comparison, we compare all the methods using learning rate warm-start. For fine-tuning RoBERTa on SNLI we used an AdamW optimizer and a learning rate of 1e-5 for 8 epochs, with mini-batch size 32. We ran all experiments with a single NVIDIA RTX A6000 GPU.



(a) CIFAR-10 (b) CIFAR-100 (c) TinyImageNet (d) SNLI

Figure 2.2: Normalized run-time and test accuracy of Crest by that of full data, when training ResNet-20 on CIFAR10, ResNet-18 on CIFAR100, ResNet-50 on TinyImagenet, and fine-tuning RoBERTa on SNLI.

Evaluation. We evaluate all the methods under 10% budget for training. That is, for Craig, GradMatch, and Glister, we find a new coreset of size 10% of the full data at the beginning of *every epoch*. On the other hand, Random iteratively selects random mini-batches, and Crest finds mini-batch coresets and automatically finds the time to update them. We stop all methods after the same number of training iterations as that of 10% training on the full data. Note that under the above 'training setup', the Random baseline achieves a much higher accuracy than that of epoch 20 of a standard 200 epoch training pipeline (see SGD† in Table 2.1). This is because the learning rate drops twice during training on Random (and coresets) under 10% budget.

Crest Setup. In our experiments, we used $b = 5$, and $T_2 = 20$ for all the datasets, and tuned τ, α and h, as discussed in Appendix 9.1.2. Nevertheless, our method is not very sensitive to the choice of α. We used $|V_p| = |V_r| = r = 0.005 \times n$ for SNLI and $0.01 \times n$ for the rest of the datasets, without further tuning. For RoBERTa we used last layer gradient and Hessian diagonal, and for other networks we used full gradient and Hessian diagonal in Eq. (2.6).

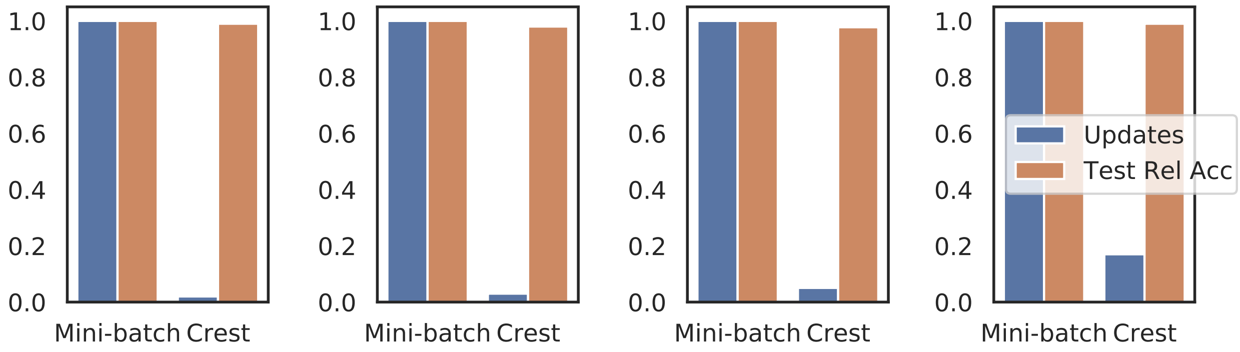
2.4.1 Evaluating Accuracy and Speedup

Accuracy. Table 2.1 shows the relative error, i.e., $\frac{|acc_{coreset} - acc_{full}|}{acc_{full}}$ of models trained with each coreset selection algorithm. We see that while the baselines yield a very high relative error in particular for larger models and more difficult tasks, e.g. CIFAR100 and TinyImageNet, Crest can successfully outperform all the baselines and obtain up to 18.2% better relative error compared to baseline coreset selection methods, and up to 2.3% better relative error compared to Random baseline. Note that as the size of the data increases, existing methods that select coresets from the full data become prohibitively expensive. Notably, CREST is the only coreset selection method that is applicable to SNLI with 570k examples. Other coreset baselines that find subsets from the full data cannot scale to such a large data. Table 2.1 confirms that CREST can successfully finds mini-batch coresets with small bias and variance and identify when they need to be updated during the training.

Speedup. Figure 2.2 compares the accuracy and wall-clock run time of CREST vs baselines, and training on full data. We see that CREST is able to achieve up to 2.5x speeds up over training on full data, while introducing the smallest relative error compared to the baselines, when training ResNet-20 on CIFAR-10, ResNet-18 on CIFAR-100, ResNet-50 on TinyImageNet, and fine-tuning RoBERTa on SNLI. Table 2.2 further lists the average wall-clock time for selecting every mini-batch coreset of size 128, calculating the quadratic loss approximation based on Eq. (2.6), and checking the validity of the approximation on a random subset of data according to Eq. (2.10), when selecting coresets with CREST to train ResNet-18 on CIFAR100. Note that selecting a mini-batch from a larger random subset is much faster than selecting a subset of size 10% from the full data, done by the baselines.

2.4.2 Ablation Study

Modeling the loss. Next, we evaluate the effectiveness of CREST in approximating the loss as piece-wise quadratic regions and identifying the time that the coresets need to be updated. To do so, we compare CREST with greedy mini-batch selection, which selects every mini-batch by applying the greedy algorithm to solve Eq. (2.5) on one random subset, and trains on it before selecting the next mini-batch. Figure 2.3 compares the relative error and the number of times CREST updates the coresets to greedy mini-batch selection. We see that CREST can effectively reduce the number of updates to 2% and 3% of the total update time of greedy mini-batch selection while preserving 98% and 99% of its performance, when training ResNet18 on CIFAR-100 and ResNet20 on CIFAR-10. For training ResNet50 on TinyImagenet and fine-tuning RoBERTa on SNLI, CREST reduces the number of updates



(a) CIFAR-10 (b) CIFAR-100 (c) TinyImageNet (d) SNLI

Figure 2.3: Normalized test accuracy and number of coreset updates for CREST over greedily selecting every mini-batch from a larger random subset by solving Eq. (2.5).

Table 2.2: Average time for different components of CREST for training ResNet-18 on CIFAR-100 with batch size 128.

Step	Time (seconds)
selection (CREST)	0.006
selection (CRAIG)	0.089
Loss approximation	0.115
Checking threshold	0.796

to 19% and 26% respectively, while preserving 95% and 99% of the performance.

Quadratic approximation. As discussed in Sec. 2.3.1, in later stages of training, the loss can be better approximated as a convex quadratic function within larger neighborhoods. Figure 2.4 (left) shows that as training proceeds, CREST can successfully increase the size of the neighborhoods in which the quadratic approximation is valid, and reduce the number of updates over time. Moreover, Figure 2.4 (right) shows that using a first-order approximation instead of our quadratic approximation, or not smoothing the gradient and curvature in calculating the quadratic approximation, leads to higher number of coreset updates, and harms the accuracy. Section 5.5.1 further compares the number of updates and the relative error at the end of training. We see that excluding the learned examples further improves the performance of CREST.

Section 5.5.1 and Figure 2.4 show that it is crucial *when* the coresets are updated. Figure 2.4 shows that updating the coreset more frequently in the beginning is the key

Table 2.3: Effect of CREST components (ResNet20/CIFAR10).

Algorithm	Rel. Error	# Updates
CREST-First	7.45	343
CREST w/o smooth	7.44	369
CREST w/o excluding	4.61	346
CREST	4.33	185

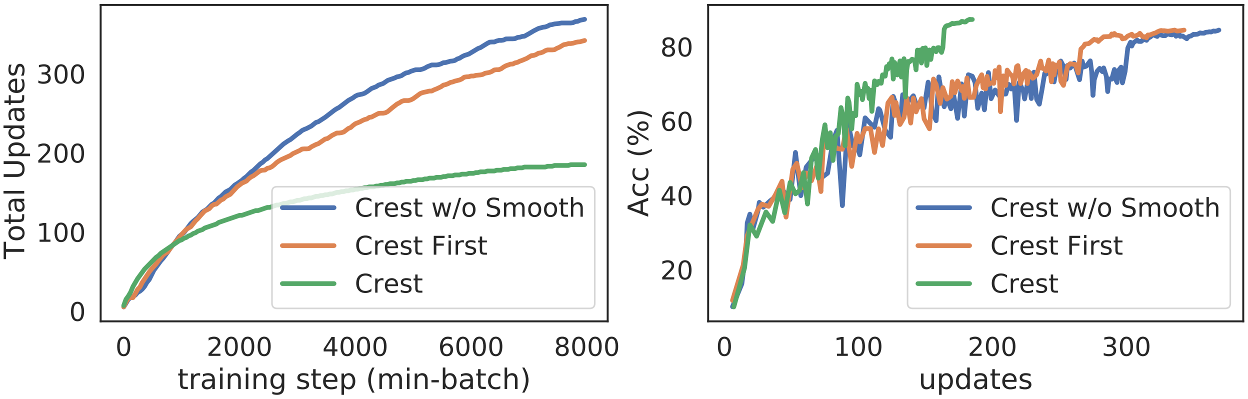


Figure 2.4: Training ResNet-20 on CIFAR-10 with CREST under 10% training budget. (Left) Number of coreset updates vs. training iterations. (Right) test accuracy vs. the total number of coreset updates.

(notice that the green line is slightly higher than blue and orange in the first 1000 iterations). This slight difference results in a much better final accuracy. However, updating the coreset frequently later in training does not improve the accuracy (it does not hurt but does not help). Hence, blue and orange lines achieve a lower accuracy than Crest with more updates. CREST can accurately find when is best to update the coresets based on its quadratic loss approximation, and achieve a better accuracy while minimizing the number of updates.

Importance of Examples during the Training. Figure 2.5 shows the average forgetting score for the selected examples during the training. Forgetting score counts the number of times examples are misclassified after being correctly classified during the training, and quantifies the difficulty of learning an example [183]. We see that CREST selects examples of increasing difficulty during the training, and excluding the learned examples allows further focusing on the difficult-to-learn examples. In contrast, random subsets have a constantly lower forgetting

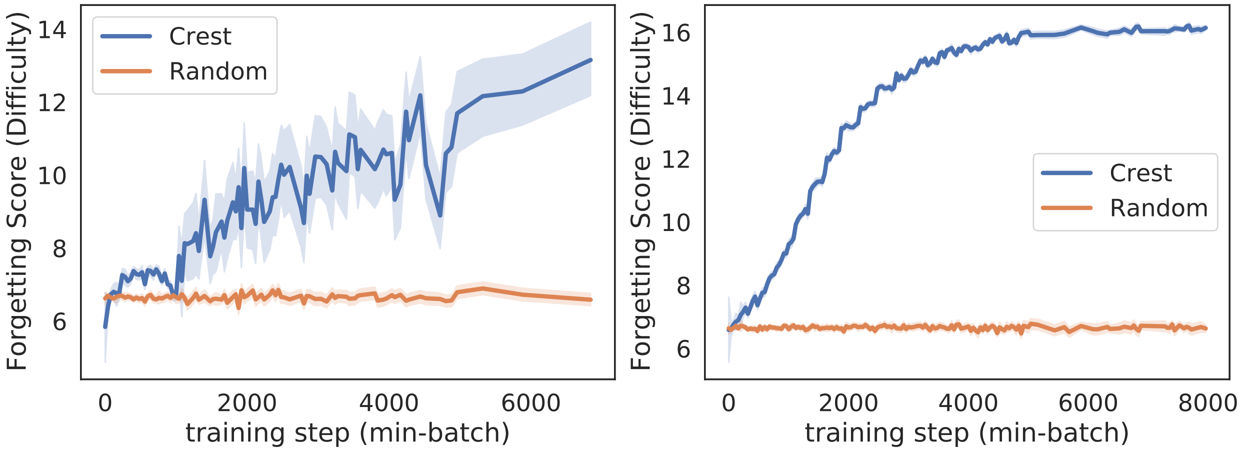


Figure 2.5: Average forgettability score of Crest coresets during training, when learned examples are not discarded (Left), and are discarded (Right). Learning difficulty of examples selected by Crest increases during the training.

score during the training. Figure 9.2b in Appendix 9.1.2 shows that while Crest trains on a diverse set of examples, the distribution of the number of times different examples are selected by Crest is very long-tailed. This shows not all examples contribute equally to training, and Crest can successfully find examples that are important for learning at different times.

Limitations. In general, coreset methods are most beneficial under a limited training budget. While Crest can still achieve a superior accuracy under a larger budget (Table 9.2 in Appendix 9.1.2), it achieves a smaller accuracy gap compared to the Random baseline. Besides, more efficient data loading can significantly speed up coreset selection.

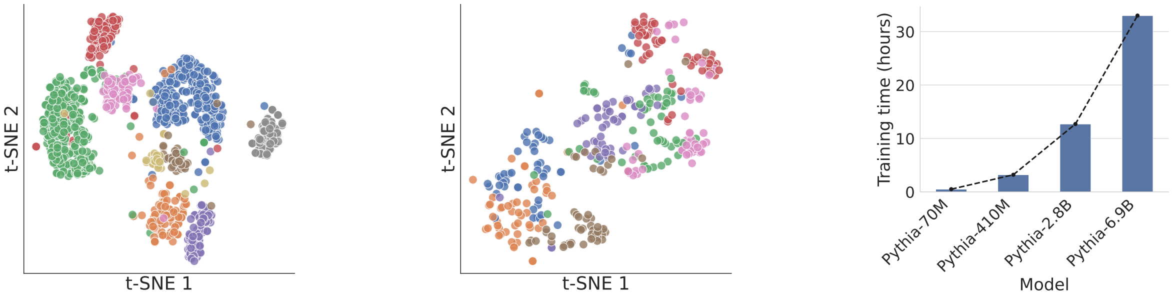
2.5 Conclusion

We proposed the first scalable framework with rigorous theoretical guarantees to identify the most valuable examples for training non-convex models, particularly deep networks. Our approach models the non-convex loss as a series of quadratic functions and extracts a coreset for each quadratic sub-region. In addition, to ensure convergence of stochastic gradient methods such as (mini-batch) SGD, it iteratively extracts multiple coresets from smaller random subsets of training data, to ensure nearly-unbiased gradient estimates with small variance. In doing so, it provides rigorous theoretical guarantee for convergence of the extracted coresets to stationary point of a non-convex function. With extensive experiments, we confirmed the effectiveness of our method on various vision and NLP deep learning tasks.

CHAPTER 3

S2L: Data-Efficient Training for Large Language Models

In recent years, large language models (LLMs) have revolutionized artificial intelligence by demonstrating an unprecedented ability to understand and generate human language [25]. Among all the contributing factors, the quality and selection of data is becoming increasingly recognized for its importance in training LLMs effectively. Recent research indicates that LLMs benefit more from training for additional epochs on carefully curated data rather than on larger, uncurated ones during pretraining [182] and instruction fine-tuning [235], making data selection one of the most promising means to unlock the next level of LLMs' language capability. However, while generalist models obtained through pre-training or instruction fine-tuning excel in *general language tasks*, they may not deliver optimal outcomes in *specialized*



(a) Hidden states of the Pile on pretrained Pythia-410M

(b) Hidden states of MathInstruct on pretrained Pythia-410M

(c) Increase in training time as the size of the model scales up

Figure 3.1: Existing data selection methods depend heavily on the feature representations from a reference model, which makes their effectiveness vulnerable to the quality of training on the target domain [118]. For supervised fine-tuning (SFT), while pretrained models can effectively separate topics (shown in different colors) in natural language (Figure 3.1a), they struggle with fine-tuning data that deviates from the pretraining distribution (Figure 3.1b). Additionally, the cost of training a reference model escalates with model size (Figure 3.1c), making existing data selection methods for large models prohibitively expensive.

domain, such as mathematics [13, 112, 223, 104, 224], code [152, 113], medicine [169, 170, 38], or finance [205, 38]. These domains are not only critical for real-world applications but also hold substantial economic and societal impacts.

To maximize performance in specialized domains, models fine-tuned on domain data offer superior capabilities over generalist models [79]. Yet, maximizing the data efficiency in supervised fine-tuning (SFT) for specialized domains remains a challenging and under-explored problem. Firstly, heuristic approaches that are effective in the instruction fine-tuning stage, like manual curation [235] or using advanced models such as GPT-4 for dataset evaluation [33], are less reliable due to the need for specialized knowledge and become costly with large volumes of uncurated fine-tuning data. Beyond these heuristic methods, other approaches rely on generating representations for each training example using a reference model, often utilizing metrics like perplexity [118], confidence [177, 192], or hidden states [2, 182, 216, 16] as features. However, these techniques also fall short in SFT for specialized domains for two reasons: (1) the significant shift between pretraining and SFT data can render these metrics less informative (Figure 3.1b), and (2) the computation and memory demands associated with generating representations for each training example become prohibitive, as these specialized domains often require larger models, some with up to 540 billion parameters [39, 169], leading to substantial scalability challenges (Figure 3.1c).

To tackle the challenges of data efficiency in SFT for specialized domains, we present SmallToLarge (S2L), an effective and scalable data selection algorithm. S2L operates by first gathering training loss trajectories for each training example using a small model. These trajectories are then clustered, and similar number of examples are selected from these clusters uniformly at random. This process is grounded in our theoretical findings that examples within the same cluster exhibit similar gradients during training, thereby affecting the model similarly. Consequently, subsets sampled from these clusters have a bounded gradient error w.r.t. the full data, allowing for training a comparable model with only a subset of data. Furthermore, we provide a convergence rate analysis for training on these subsets, establishing a robust theoretical foundation for S2L's effectiveness and efficiency.

To validate S2L's effectiveness, we applied it to the challenging tasks of SFT for (1) mathematical problem-solving and (2) clinical text summarization. Our experiments on MathInstruct [224] shows that S2L can significantly reduce the required training data size to just 11% of the original dataset size while still matching the performance levels of the full dataset, outperforming current state-of-the-art one-shot and online data selection algorithms by an average of 4.7% across 6 in- and out-domain evaluation datasets. Remarkably, on the MATH benchmark [68], S2L attained a 32.7% accuracy with just 50K data points, improving the best open-sourced model under 3 billion parameters, Phi-2, by 16.6%. For clinical text summarization tasks on the MIMIC-III [83] dataset, S2L outperforms training on the full dataset, using only half of the data. Unlike existing methods that require training and getting features from large models, S2L achieves superior data efficiency using a model with as few as 70 million parameters, which is 100× smaller than the largest target model we train with 7 billion parameters.

3.1 Related Work

Foundations of Data Selection. Data selection has been well studied for small models and classification tasks. There are one-shot algorithms that select data based on rankings of the proposed training statistics, for example, the L2-norms of error and gradient vectors (EL2N and GraNd) [143], confidence and its variability across epochs [177], and the number of times each example is learned but then forgot at the subsequent training step [184]. Besides these heuristic indicators, there are embedding-based pruning algorithms [173] and online selection algorithms with theoretical performance guarantees for efficiency [127, 88, 89, 147, 214] and robustness [213, 217, 52]. **(author?)** proposed to use the intermediate feature representation of a small proxy model to select data for image classification. Most recently, data selection has shown great potential in more substantial training speedup when implemented on near-storage hardware [149], and data selection beyond supervised learning of image data, e.g., for self-supervised learning [84] and multimodal learning [2, 116], also emerged.

Data Efficient Training of Large Language Models. For the pre-training of LLMs,

(author?) studied data quality indicators including Perplexity, Error L2-Norm (EL2N) [143], and memorization ranking [17], and found training on examples with middle Perplexity rankings outperforms training on examples selected based on the other two metrics, and sometimes even outperforms training on the entire dataset. **(author?)** uses pre-trained model embeddings to select data for LLM pre-training. The proposed algorithm, D4, first applies an embedding-based data de-duplication algorithm [2] and then discards data points that are the closest to the K-Means cluster centroids in the embedding space [173] to ensure diversity. On fine-tuning LLMs, existing work on data efficiency primarily focused on manually curating high-quality instructions [235], or using strong closed-source models (e.g., GPT-4 [3] or ChatGPT) to rate the quality of each training example [56, 101, 33]. **(author?)** implemented an experimental design framework to evaluate the existing data selection methods for instruction fine-tuning of LLMs and found selecting facility locations based on hidden representations (i.e., embeddings) is the most effective. As the only data selection algorithm for specialized domains, SCIP [216] focuses on pruning low-quality code data for training code LLMs. Since it relies on breaking the code syntax to understand the characteristics of low-quality code in the embedding (i.e, hidden states) space, adapting SCIP to domains other than Python code data is non-trivial.

Adapting Large Language Models for Specialized Domains. The rapid development of large language models (LLMs) gives rise to new state-of-the-art models in specialized domains. For mathematical reasoning, Galactica [180], MINERVA [98] and Llemma [13] continue to train an LLM on large-scale math-related web data to improve a model's general scientific reasoning; WizardMath [112] and TinyGSM [104] fine-tune LLMs using supervised data. Similarly for medical LLMs, **(author?)** continued training pre-trained LLMs on medical text, and [169, 170] fine-tuned PaLM with instruction prompt tuning on medical domain data.

3.2 Problem Formulation

LLM Fine-tuning Objective. Consider a transformer-based language model, parameterized by $\boldsymbol{\theta}$, and denoted as $p_{\boldsymbol{\theta}}$. This model, when provided with a sequence of prompt tokens $\mathbf{x} = (x_1, \ldots, x_M)$, generates a sequence of response tokens $\mathbf{y} = (y_1, \ldots, y_L)$. The conditional probability of generating $\mathbf{y}$ given $\mathbf{x}$ is then formulated as

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{L} p_{\boldsymbol{\theta}}(y_l|\mathbf{y}_{1:l-1}, \mathbf{x}). \tag{3.1}$$

Note that $\mathbf{y}_{1:0}$ is an empty sequence. To adapt the pre-trained LLM for a specialized domain of distribution $\mathcal{D}$, supervised fine-tuning (SFT) is usually employed with a domain-specific training dataset $D_{\text{train}} = \{(\mathbf{x}, \mathbf{y})_i\}_{i=1}^{n} \sim \mathcal{D}$ containing pairs of prompt $\mathbf{x}$ and annotated response $\mathbf{y}$. The fine-tuning objective is thus to minimize the following negative log likelihood loss, expressed as:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, D_{\text{train}}) = -\frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})_i \in D_{\text{train}}} \big[\log p_{\boldsymbol{\theta}}(\mathbf{y}_i|\mathbf{x}_i)\big]. \tag{3.2}$$

Data Selection Objective. In a general setting for data selection, we consider a target language model $p_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$. Given a fixed data budget B, which constrains the number of data points that can be used for training, our objective is to select a subset $S \subseteq D_{\text{train}}$ to train the target model, such that it obtains a superior generalization performance. In practice, the subset S is selected based on a reference model $r_{\boldsymbol{\phi}}$ parameterized by $\boldsymbol{\phi}$, which generates representations, confidence scores, or other metrics for each data point $(\mathbf{x}, \mathbf{y})_i \in D_{\text{train}}$, denoted by $r_{\boldsymbol{\phi}}((\mathbf{x}, \mathbf{y})_i)$, which will be utilized by a data selection algorithm to produce S.

In existing data selection algorithms, $\boldsymbol{\phi}$ is commonly either weights of the pre-trained target model or a target model that has been fully trained on the dataset D_{train}. However, as evidenced by Figure 3.1, representations generated by the pretrained model may not always be good enough for data selection in specialized domains, and fine-tuning the target model significantly increases the computational cost of data selection.

3.3 Methodology

Training a large target model to obtain feature representations for each example in D_{train} can be computationally intensive. However, a recent finding demonstrates that the training dynamics of most examples are consistent across differently sized models of the same family, and this phenomena even generalizes across different model families [206]. Our proposed method, **S**MALL**T**O**L**ARGE **(S2L)**, leverages *loss trajectories* of training examples collected during fine-tuning a *small* reference model on the full or a subset of training data.

Loss Trajectory. Let $\boldsymbol{\phi}^{(t)}$ be the parameters of a small LM during training on D_{train} at times $t_q, q \in \{1, ..., T\}$. S2L records the loss trajectory for each data point i at times t_q during training the reference model $[\mathcal{L}_i^{\text{proxy}}(\boldsymbol{\phi}^{(t_1)}), \ldots, \mathcal{L}_i^{\text{proxy}}(\boldsymbol{\phi}^{(t_T)})]$ where

$$\mathcal{L}_i^{\text{proxy}}(\boldsymbol{\phi}^{(t)}) = \mathcal{L}^{\text{proxy}}(\boldsymbol{\phi}^{(t)}, (\mathbf{x}_i, \mathbf{y}_i)) = -\log p_{\boldsymbol{\phi}^{(t)}}(\mathbf{y}_i|\mathbf{x}_i), \tag{3.3}$$

and T is the length of the loss trajectory. Note that $\boldsymbol{\phi}^{(t)}$ is trained for a fixed number of iterations from $\boldsymbol{\phi}^{(t-1)}$.

Assume the parameter vector $\boldsymbol{\theta}^{(t)}$ represents the parameters of the target model at the time t. Define $\mathbf{L}_i^{\text{proxy}} = [\mathcal{L}_i^{\text{proxy}}(\boldsymbol{\phi}^{(t_1)}), \ldots, \mathcal{L}_i^{\text{proxy}}(\boldsymbol{\phi}^{(t_T)})]$ and $\mathbf{L}_i^{\text{target}} = [\mathcal{L}_i^{\text{target}}(\boldsymbol{\theta}^{(t_1)}), \ldots, \mathcal{L}_i^{\text{target}}(\boldsymbol{\theta}^{(t_T)})]$ as the training loss trajectory of the example i on the small proxy model and the large target model, respectively. Let $\boldsymbol{H}_i \in \mathbb{R}^{d \times d}$ be the Hessian matrix for each example i and assume that the loss function for each example during fine-tuning can be modeled by a second-order Taylor approximation with bounded curvature ($c \leq \|\boldsymbol{H}_i\| \leq C$), a reasonable assumption in fine-tuning settings. The following lemma shows that examples with similar loss trajectories on the proxy model have similar gradients throughout the training of the target model.

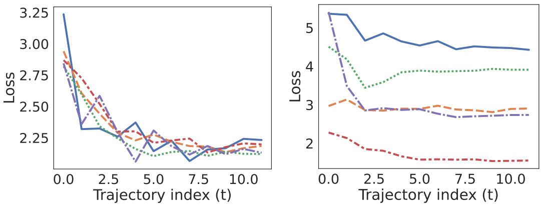
Theorem 3.3.1. *If examples i and j have similar loss trajectories on the proxy model, i.e., $\|\mathbf{L}_i^{proxy} - \mathbf{L}_j^{proxy}\| \leq \epsilon$, and their loss trajectories on the proxy and target model is similar, i.e.,*

$\|\mathbf{L}_p^{proxy} - \mathbf{L}_p^{target}\| \leq \delta$ *for* $p \in \{i, j\}$, *then* i *and* j *have similar gradients throughout training the target model:*

$$\|\nabla \mathcal{L}_i^{target}(\boldsymbol{\theta}) - \nabla \mathcal{L}_j^{target}(\boldsymbol{\theta})\| \leq \frac{2\epsilon' + 2CD^2}{d} = \Delta. \tag{3.4}$$

where $\epsilon' = \epsilon + 2\delta$ *and* $\|\boldsymbol{\theta}\| \leq D$ *for all* t.

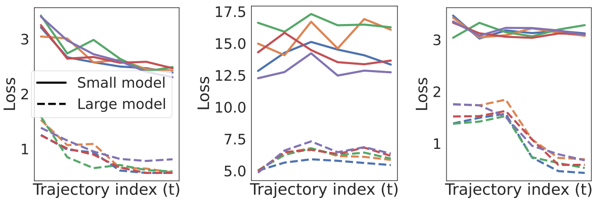
The proof of Theorem 3.3.1 can be found in Section 9.2.1.1. Theorem 3.3.1 shows that examples with similar loss trajectories have similar gradients during the training, thereby influencing the model in a similar manner.



(a) In the same cluster.

(b) In different clusters.

Figure 3.2: Examples in the same clusters have very similar loss trajectories (Figure 3.2a) while the loss trajectories of examples in different clusters are very different (Figure 3.2b).



(a)

(b)

(c)

Figure 3.3: Examples in the same clusters of training trajectories on a small model (Pythia-70M) also have similar training trajectories on a large model (Pythia-2.8B), even if the trends may not be the same on both models.

Data selection from Loss Trajectory Clusters. Once the loss trajectories are recorded on the proxy model, we apply a clustering algorithm to group examples based on the similarity of their loss trajectories. This results in a set of clusters $\{C_1, C_2, \ldots, C_K\}$, where each cluster C_i contains examples with similar loss and gradient trajectory throughout the training:

$$C_i = \{(\mathbf{x}, \mathbf{y})_j \in D_{\text{train}} | i = \arg\min_{j \in [K]} d(\mathbf{L}_j, \mathbf{L}_{\bar{C}_j})\}, \tag{3.5}$$

where $\mathbf{L}_{\bar{C}_i}$ is the centroid of the loss trajectories in cluster C_i, and $d(\cdot, \cdot)$ is a distance metric, such as Euclidean distance, used for clustering. For datasets that contain different sources of data, we cluster each source separately.

Algorithm 2 Data Selection Based on Training Trajectories (S2L)

```
Require: Training dataset $D_{\text{train}}$ with corresponding training trajectories, a fixed data budget $B$, number of clusters $K$.
Ensure: Subset $S \subseteq D_{\text{train}}, |S| \leq B$.
Initialize $S$ as an empty set.
Train a small proxy model and cluster examples in (each data source of) $D_{\text{train}}$ based on their loss trajectories and sort them by size to get $\mathcal{C} = \{C_1, C_2, \dots, C_K\}$.
for each cluster $C_k$ in $\mathcal{C}$ do
    Calculate $R_k$, the number of examples to randomly sample from $C_k$, i.e., $R_k = (B - |S|)/(K - k + 1)$.
    if $|C_k| \leq R_k$ then
        $S \leftarrow \{S \bigcup C_k\}$.
    else
        $S \leftarrow \{S \bigcup S_k\}$, where $S_k \subset C_k$ is selected uniformly at random from $C_k$ and $|S_k| = R_k$
    end if
end for
Return $S$
```

As shown in Figure 3.2, clustering algorithms can effectively find groups of examples with similar training dynamics. In Figure 3.3, we empirically show that we can identify groups of examples with similar training dynamics on a larger model by clustering the training trajectories of D_{train} on a smaller proxy model. With the clusters formed, the data selection strategy selects equal number of examples at random from all clusters, as detailed in Algorithm 2. In doing so, it effectively prioritizes selecting examples from smaller clusters. This is particularly important for datasets containing multiple imbalanced sources. In this setting, training and test distributions often differ, and balanced selection from clusters ensures superior test performance on all groups in the test data.

The following theorem shows that, under the assumptions of Theorem 3.3.1, training with Incremental Gradient (IG) methods on the subset selected by S2L converges to a close neighborhood of the optimal solution found by training the target model on the full dataset. IG methods such as Stochastic Gradient Descent (SGD) update parameters iteratively based on the gradient of the loss of individual examples, multiplied by stepsize α. Formally,

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \alpha \nabla \mathcal{L}_i^{\text{target}}(\boldsymbol{\theta}^{t}). \tag{3.6}$$

Corollary 3.3.2. *Under the assumptions of Theorem 3.3.1, applying IG with stepsize α to subsets found by S2L, converges to the neighborhood of the optimal solution, as follows:*

$$\|\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^*\|^2 \leq (1 - \alpha c)^{t+1}\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*\|^2 + 2\xi R/c^2 + \alpha B^2 (r_{\min}/k)^2 \boldsymbol{g}_{\max}^2 \tag{3.7}$$

where $c \leq \|\boldsymbol{H}\|$, $B = k \cdot K$ is the total size of the subset, $\boldsymbol{g}_{\max}$ is the largest gradient norm of individual examples during training, $r_{\min} = \min_j |C_j|, r_{\max} = \max_j |C_j|$, $R = \min\{d_0, B\boldsymbol{g}_{\max} + \xi/c\}$ and $d_0 = \|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^*\|$ is the initial distance to the optimal solution $\boldsymbol{\theta}^*$, and ξ is given by:

$$\xi = K[r_{\min}\Delta + (r_{\max} - r_{\min})\boldsymbol{g}_{\max}]. \tag{3.8}$$

The proof can be found in Section 9.2.1.2.

3.4 Experiments

In this section, we present the comprehensive experiments conducted to evaluate the efficacy of the proposed data selection method, SmallToLarge (S2L), across two challenging domains (mathematical reasoning and clinical text summarization).

3.4.1 Baselines

We systematically compare S2L against a comprehensive set of open-sourced data selection methods. These methods are categorized based on the type of representation they use and selected as the most representative or best-performing methods as identified in prior work. These include: (1) **Random Sampling**; selecting examples with the (2) **Least Confidence** [16] or (3)**Middle Perplexity** [118]; (4) **High Learnability**, determined by the loss decrease before and after full fine-tuning [236]; and (5) **Facility Locations** selection based on hidden states [16]. Additionally, we incorporate one online selection techniques: (6) **Confidence Curriculum** proposed by **(author?)**, which selects examples with decreasing confidence during the training. Given that the optimal reference model may vary for each one-shot selection method, we ensure a fair comparison by adopting the approach used in [118], which runs each method with both the fully fine-tuned target model and an early fine-tuning checkpoint as the reference model. We report the best results from these setups.

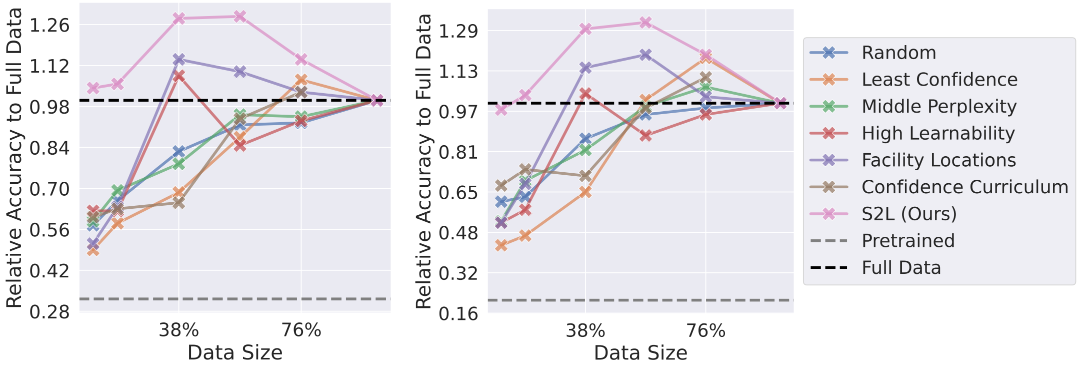
3.4.2 Specialized Domain 1: Mathematical Reasoning

Training Settings. We focus on fine-tuning using the **MathInstruct** dataset [224] with 262,040 training examples for its comprehensive coverage of diverse mathematical fields and its capability in training models to achieve state-of-the-art performance on the standard evaluation benchmarks. We employ the open-source model suites Pythia [18], Phi-2 [102], Llama-2 [185] as our base models to validate our S2L algorithm and directly compare its performance against the state-of-the-art.

Evaluation Datasets. We follow the framework established in [224] for a comprehensive assessment using several well-regarded datasets, including in-domain and out-of-domain datasets. For the in-domain datasets, we consider **GSM8K** [41], **MATH** [68], and **NumGLUE** [128]. For the out-of-domain datasets, we consider **SVAMP** [142], **Mathematics** [46], **SimulEq** [94]. These datasets collectively span a diverse range of mathematical subjects, such as algebra, probability, number theory, calculus, and geometry. Additionally, some questions in these datasets require the application of commonsense, reading comprehension, and multi-step reasoning. All questions are open-formed.

Evaluation Metric. We use the standard evaluation metric for open-formed questions, **exact match**, which measures the model's accuracy by comparing its generated answers against the correct solutions. For an answer to be considered correct, it must match the reference solution precisely.

More details about the settings and baseline implementations can be found in Section 9.2.2.



(a) In-domain Avg

(b) Avg

Figure 3.4: **Data Scaling:** Accuracies (↑) on in-domain and out-of-domain datasets using Pythia-410M. Data size refers to the total number of unique training examples used. All experiments in this table share the same total training steps and learning rate schedule (see Section 3.4.2). See breakdowns in Figure 9.5.

Table 3.1: **Less Data, Same Compute:** Zero-shot accuracies (%, ↑) when S2L and the baselines select 50K data to train with the same number of iterations as the full-data training. Results surpassing full training are highlighted in bold. Figure 3.4 follows the same setting but uses the Pythia-410M model.

Target	Fine-tuning	In-domain			Out-domain			
Model	Data	GSM8K	MATH	NumGLUE	SVAMP	Mathematics	SimulEq	**Avg**
Phi-2 (2.7B)	(Pretrained)	53.4	16.1	34.9	67.9	31.1	27.4	38.5
	Random	67.9	30.1	60.7	77.1	51.2	37.5	54.1
	High Learnability	59.4	25.2	62.1	76.6	41.8	27.2	48.7
	Middle Perplexity	66.4	29.5	54.1	74.8	50.4	39.8	52.5
	Least confidence	61.7	24.7	67.0	76.5	43.3	52.5	54.3
	Facility Locations	66.2	31.3	62.4	74.4	58.4	34.6	54.5
	S2L(Ours)	**69.1**	**32.6**	**65.7**	**79.6**	56.4	40.1	57.3
	Full-262K	68.3	32.6	64.3	78.4	58.4	44.2	**57.7**

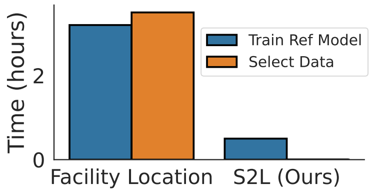


Figure 3.5: Wall-clock time required to train the reference model and select 100K data from MathInstruct for training Pythia-410M.

3.4.2.1 Setting 1: Less Data for Better Models

In the first setting, we standardize the number of training steps to correspond to 3 epochs on the full dataset, aligning with [224]. This allows us to maintain a consistent training schedule across different methods and data budgets, ensuring fair and accurate comparisons of the quality of data.

Scaling the Data: SOTA algorithms fail with small data budgets while S2L stands out across data scales. In Figure 3.4, we compare S2L against the baselines from Section 3.4.1 on Pythia-410M across varying data sizes. The training trajectories used by S2L are based on Pythia-70M, a model approximately 6x smaller than Pythia-410M. With the same number of training steps as the full training, S2L exceeds the full dataset's performance using only 30K examples, only 11% of the full dataset. It leads the runner-up baselines by an average of 4.7%, 4.6% and 2.4% with data budget 30K, 50K and 100K across all six evaluation datasets. While state-of-the-art data selection algorithms like Facility Locations [16] and High Learnability [236] have decent performance with a large enough data budget (i.e., 100K), all SOTA algorithms except S2L cannot even outperform the random sampling baseline when the allowed data size is small (i.e., 30K). Unlike the existing algorithms, S2L consistently outperforms all baselines and even full training across all data sizes. Note that compared to the runner-up algorithm in this setting, Facility Locations, the cost of S2L is much lower in both training the reference model and data selection stages (Figure 3.5), and therefore more scalable to both larger target models or larger data sizes.

Scaling the Model: Data selected by S2L can transfer to larger models in different model suites. We also test whether this subset, chosen using Pythia-70M, can effectively train larger models beyond 410M and models outside the Pythia suite. As shown in Table 3.1, our experiments with Phi-2 reveal that fine-tuning on only 50K S2L-selected data again outperforms full dataset training on the most challenging MATH [68] benchmark improving the pretrained Phi-2 by 16.6% and is more data efficient than training on the full MathInstruct dataset to get the same performance.

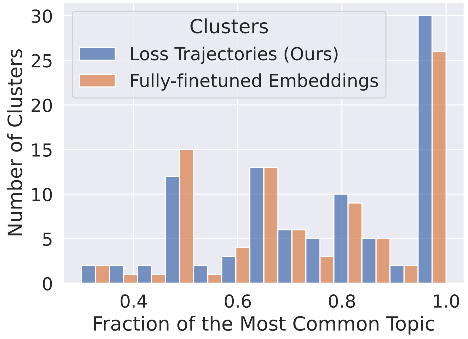


Figure 3.6: Distribution of the coverage of top-1 topic in each cluster. Taller bars on the right end of the plot indicate clusters with a higher concentration of a single topic and therefore suggest a grouping of similar examples.

Table 3.2: **Less Data, Same Epochs:** Zero-shot accuracies (%, ↑) when S2L trains 50% data for the same number of epochs as the full-data training. S2L can achieve comparable performance to full-data training while reducing both the data storage space and the training time by half.

Target Model	Fine-tuning Data	In-domain GSM8K	In-domain MATH	In-domain NumGLUE	Out-domain SVAMP	Out-domain Mathematics	Out-domain SimulEq	**Avg**
Phi-3-mini (3.8B)	(Pretrained)	74.5	26.5	52.1	83.7	44.3	34.8	52.7
	S2L-50%(Ours)	76.3	42.5	76.4	83.8	62.1	51.6	65.4
	Full	76.4	42.9	75.3	84.6	60.2	51.9	65.2
Llama-2-7B	(Pretrained)	3.1	4.2	16.5	14.1	8.3	2.3	8.1
	S2L-50%(Ours)	53.3	28.9	65.0	65.1	45.2	31.9	48.2
	Full-262K [224]	52.2	30.4	60.5	65.3	43.9	50.2	50.4

3.4.2.2 Setting 2: Less Data for Faster Training

The second setting we consider is when fixing the number of times each example can be seen over the entire course of training, directly translating smaller datasets into reduced training and storage costs. This is particularly beneficial for large models that would require extensive training times without data selection. By experimenting with models of larger sizes than the previous setting, we observe in Table 3.2 that S2L can achieve comparable performance to full-data training when using only 50% data and thereby reducing both the data storage space and the training time by half.

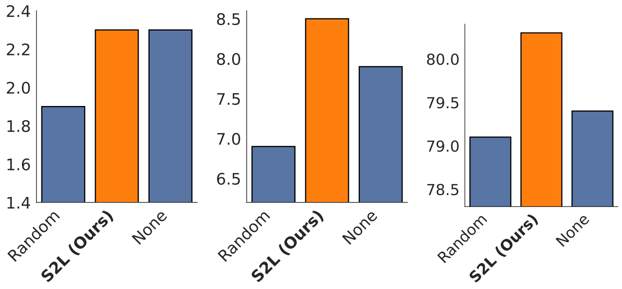
3.4.2.3 Why is S2L So Effective?

Examples in Clusters Encode the Same Knowledge/Skill. In Section 9.2.3, we compare actual training examples in MathInstruct that get clustered together due to their similar training trajectories on the small Pythia-70M model. We observe that examples in the same cluster are of the same type and related to the same knowledge/skill, e.g., open-formed algebra questions (Figure 9.6), examples requiring extracting useful information from long text and writing programs (Figure 9.7), and multiple choice questions that require multi-step reasoning (Figure 9.8), etc. Therefore, by sampling from different clusters, we make sure the selected examples cover the knowledge required for all topics and skills required for all types of questions.

Loss Trajectories can Capture the Similarity Between Data Points As Much As Embeddings of a Fully Fine-tuned Model. We conducted a quantitative analysis to assess how effectively S2L identifies similar examples using loss trajectories from a small model. Assuming math problems under the same topic require similar knowledge and share question formats, we used unknown topic labels during S2L's data selection to check if each cluster predominantly contains a single topic. By calculating the fraction of the most common topic in each cluster and plotting this in Figure 3.6 (with K=100, excluding clusters of size one), we compared the loss trajectory clusters from S2L (in blue) against those from the embeddings of a fully fine-tuned Phi-2 model (in orange)—considered the ground truth for similarity. Results show that most clusters formed by S2L using the Pythia-70M model are based on a single topic and capture topic similarities more effectively than those from the Phi-2 model's embeddings. This analysis not only confirms the homogeneity within S2L clusters but also highlights the computational efficiency of using loss trajectories on small models to identify representative examples.

3.4.3 Specialized Domain 2: Clinical Text Summarization

S2L can improve data efficiency not only for fine-tuning data not only in mathematics but



(a) BLEU (b) ROUGE-L (c) BERTScore

Figure 3.7: Performance (↑) of models trained on (1) **random**ly selected 30K examples, (2) **S2L** selected 30K examples, and (3) full 61K examples (**none**) evaluated with 3 different metrics. The minimum value on the y-axis is the performance of the model before fine-tuning. S2L improves the data efficiency for the clinical text summarization task by outperforming training on the full dataset with only less than half of the data.

also in other specialized domains. This subsection explores its application to clinical text summarization within radiology reports. This task involves processing the detailed analysis and results listed in the findings section of a radiology report and distilling them into a concise impression section. Such summaries are crucial for providing clinicians with quick and actionable insights from radiological studies.

Dataset & Setup. We use the **MIMIC-III** dataset [83], a comprehensive collection of radiology reports and findings authored by attending physicians in routine clinical care. We use the same preprocessing procedures as [48, 50] to extract the findings and impression sections and remove invalid reports. Given that access to MIMIC-III requires specific credentials, we provide a synthetic example of a radiology report generated by GPT-4 [3] for illustrative purposes in Table 9.4. We employ the Pythia-1B model and keep the training setting consistent with the mathematical reasoning task.

Evaluation. Our evaluation of generated clinical summaries on the MIMIC-III dataset's test split employs three key metrics as recommended in [191, 188]: (1) **BLEU** [140], which measures word sequence overlap between the generated and reference texts; (2) **ROUGE-L** [103], assessing the longest common word sequence; and (3) **BERTScore** [231], evaluating semantic similarity using BERT's contextual embeddings. These metrics together offer a comprehensive evaluation, ensuring our summaries are not only precise in language but also meaningful and coherent in the context of clinical information. We compare S2L to random selection, a surprisingly strong baseline as evidenced in Section 3.4.2, to check the validity of the data selection problem on this dataset and then compare it to training on the full dataset to assess its effectiveness.

Results. We compare using 30K examples selected by random vs. selected through S2L. Even with only half of the data, the model trained with S2L selected data achieves similar BLEU and significantly higher ROUGE-L and BERTSCore compared to the model trained on the entire 61.5K data. Meanwhile, training on randomly selected 30K examples performs worse than training on the full dataset on all 3 metrics. These results together demonstrate S2L's effectiveness.

3.4.4 Ablation Studies

We conduct ablation studies on MathInstruct and Pythia-410M to further understand the best practices for using S2L.

S2L is robust w.r.t. the length of the trajectories but can benefit more from longer trajectories. Figure 3.8 compares models trained with data selected by S2L based on training trajectories of different lengths. The shorter trajectories are derived from a uniform sample of the longer trajectories. From the small slopes of the lines, we can conclude that S2L is relatively robust to the length of the training trajectories. Nevertheless, we can also observe a slight increase in the performance on some of the datasets when longer trajectories are used, so having longer trajectories is still preferred.

S2L can utilize training trajectories collected at any stage of training but preferably denser ones. With the length of the trajectories fixed to 4, we can observe in Figure 3.9 that denser trajectories recorded at any training stage (early, middle, or late) are more helpful for S2L than trajectories recorded sparsely throughout the training.

S2L does not require the full training data to train the proxy and can scale

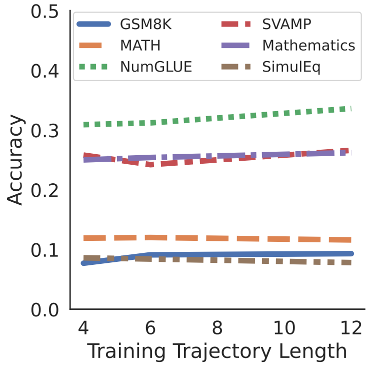


Figure 3.8: S2L is robust to the length of training trajectories.

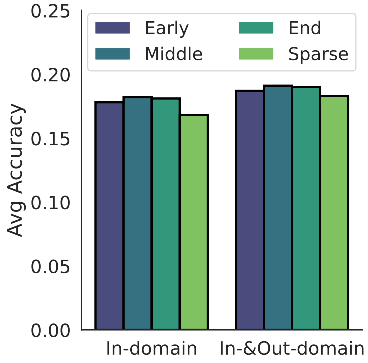


Figure 3.9: S2L prefers dense trajectories over sparse ones.

efficiently to larger datasets. To further demonstrate the scalability of the proposed S2L method, we conducted experiments by training the proxy on a smaller sample of the data (100K examples) for the same number of epochs (3 epochs) and saving the loss for all examples. The results, shown in Figure 3.10, confirm that S2L remains effective when the proxy model is trained on a smaller subset of training data and therefore is scalable to larger datasets without a proportional increase in computational costs.

S2L is robust across different clustering parameter values for K. We conducted detailed experiments varying the clustering parameter K, as shown in Figure 3.11. The results demonstrate that S2L maintains high performance across different values of K, highlighting the robustness of our method to different clustering parameter choices. We chose K=100 for our experiments as it provided the best average accuracy across the evaluation datasets for the math reasoning task.

S2L remains effective and efficient compared to using full data when trained for the same number of epochs. Figure 3.12 illustrates the relative accuracy to full data across different epochs, comparing S2L-selected data and full data with the same number of epochs. Both in-domain and overall average accuracy are shown. S2L demonstrates superior performance with fewer data and fewer training iterations.

S2L supports a range of small models as effective proxies. To understand whether different small models could serve as effective proxies, we used GPT-2 (124M) and Pythia-160M as proxy models for data selection to train Pythia-410M. The results, illustrated in Figure 3.13, show that both proxy models perform comparably in guiding the data selection, demonstrating the versatility and effectiveness of using different small models for S2L.

3.5 Conclusion and Limitations

In this work, we introduced SMALLTOLARGE (S2L), a scalable data selection method to improve the data efficiency of supervised fine-tuning (SFT) for large language models (LLMs) in specialized domains. By clustering data points based on their training dynamics on smaller

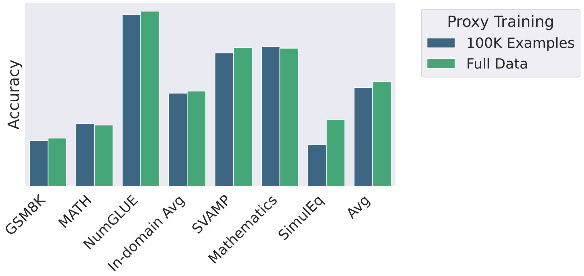


Figure 3.10: Per-dataset and average accuracy comparing proxy training on 100K examples and full data. S2L remains effective.

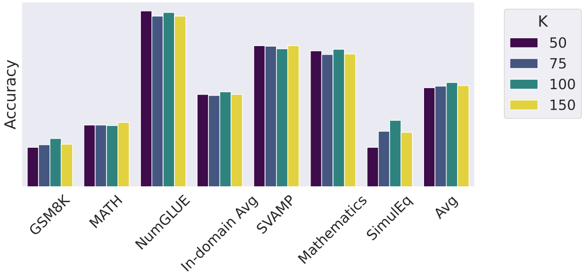
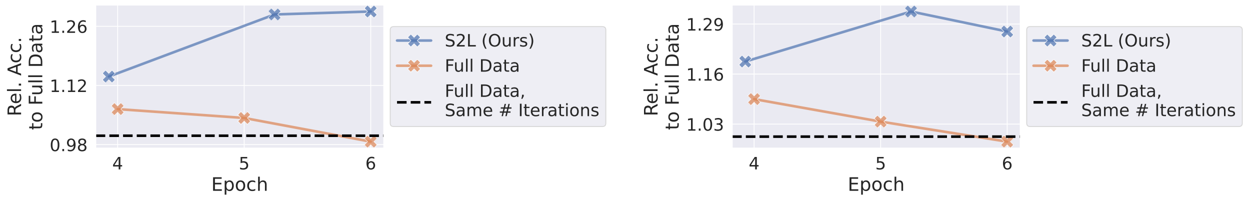


Figure 3.11: Per-dataset and average accuracy across different values of the clustering parameter K. S2L is relatively robust to the choice of K.



(a) In-domain Average Accuracy

(b) Overall Average Accuracy

Figure 3.12: Relative accuracy to full data across different epochs, comparing S2L-selected data and full data. S2L achieves superior performance with fewer data and fewer training iterations.

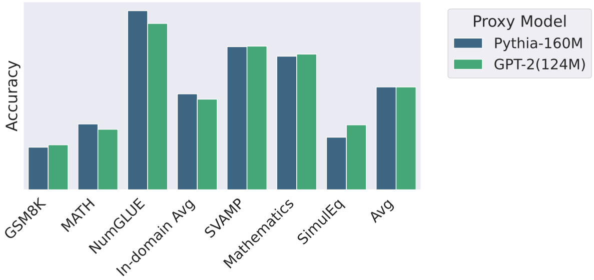


Figure 3.13: Per-dataset and average accuracy comparison between using different proxy models (Pythia-160M and GPT-2 (124M)) for data selection. Using both proxy models show comparable performance, demonstrating the effectiveness of different small models as reference models for S2L.

models and balanced sampling from all clusters, S2L significantly reduces the required training data size without compromising performance compared to using the entire training dataset. Our comprehensive experiments across the mathematical problem-solving and clinical text summarization domains demonstrate the effectiveness of S2L.

Our study does come with its limitations. S2L has been only tested within two domains, mathematics and medicine, and on models up to 7 billion parameters, constrained by our computational resources. Additionally, our experiments employed a fixed training schedule across all methods without further optimization or hyperparameter tuning for each method, including S2L. This unified approach, while it ensures a fair comparison, may not fully capture the potential performance improvement that could be achieved with more tailored training strategies. We encourage further research to extend the application of S2L across a broader spectrum of domains and investigate the impact of hyperparameter tuning on its effectiveness.

Part II

Data Selection for Robust Training against Data Poisoning

CHAPTER 4

EPIC: Robust Training Against Data Poisoning

The impressive success of modern machine learning systems is highly dependent on the quality of their large training data. Many large datasets are scraped from the internet, or other public and user-provided sources. Models trained on such datasets are susceptible to data poisoning attacks, wherein an adversary places specially-constructed poisoned examples into the training data with the intention of manipulating the behavior of the system at test time. These attacks create security vulnerabilities that cannot be detected even if the data is labeled and checked by human supervision. This makes data poisoning arguably one of the most concerning threats to deep learning systems deployed in security- and safety-critical applications, such as financial services, security cameras, autonomous cars, and medical devices.

Various types of poisoning attacks have been proposed in recent years. Most attacks fall into one of two main categories: backdoor or triggerless poisoning. Backdoor data poisoning augments the training data by a set of poisoned examples that contain a (not necessarily visible) trigger pattern [64, 189, 174]. Finetuning the model on the augmented training data causes a model to misclassify test-time samples containing the trigger. On the other hand, triggerless poisoning attacks work by crafting small per-example perturbations so that the perturbed training examples collide with the adversarially labeled target in the feature or gradient space [163, 238, 74, 60, 5]. Triggerless poisoning attacks cause misclassification of particular instances and do not require modification at inference time. In both cases, the poisoned examples may be seemingly innocent and properly labeled, and hence are hard to be detected by expert observers.

Existing defense mechanisms against data poisoning attacks mainly rely on either anomaly detection based on nearest neighbors, training loss, singular-value decomposition, feature and activation clustering [44, 175, 186, 32, 144], or robust training based on strong data augmentation, randomized smoothing, ensembling, and adversarial training [198, 97, 1, 114, 100, 179]. However, such methods either drastically degrade the generalization performance of the model [59], or can only protect the model against certain types of poisoning attacks [91, 186], or are computationally prohibitive for standard deep learning pipelines [59]. Importantly, these methods do not provide any theoretical guarantee for the performance of the model [198, 97, 1, 59].

We develop an efficient and principled defense framework that effectively prevents various types of targeted poisoning attacks, and provide theoretical guarantee for the performance of the model. To successfully prevent poisoning attacks, we make the following key observation: not all poisons are effective in making the attack successful. In particular, targeted attacks add bounded perturbations to randomly selected subsets of training data to match the gradient of the adversarially labeled target. We show that for a poison to be effective, it needs to fall close enough to the target in the gradient space. However, under bounded perturbations, only a small number of poisons can be optimized to get close enough to the target and make the attack successful. Such effective poisons get far away from their original class and get isolated in the gradient space. Eliminating the *effective poisons* can successfully break various types of attacks.

To prevent data poisoning while maintaining the generalization performance of the network, we aim to identify and eliminate the effective poisons. We show that effective poisons can be identified as isolated *medoids* of each class, in the gradient space. Medoids are the most centrally located examples of a dataset, that minimize the sum of dissimilarity between every data point to its nearest medoid. The set of medoids can be efficiently extracted by maximizing a submodular function. To eliminate effective poisons, we iteratively find medoids of every class in the gradient space during the training. Then, we assign every data point to the closest medoid in its class, and drop the medoids to which no other data point is assigned. We show that our Effective Poison IdentifiCation (EPIc) method can successfully eliminate effective poisons. We also prove that training on large gradient clusters of each class guarantees similar training dynamics to that of training on the full data.

Compared to existing defense strategies, our method does not require a pre-trained clean model, is not attack-specific, can be applied very efficiently during the training, and provides a quality guarantee for the performance of the trained model. Our extensive experiments show that our method renders state-of-the-art targeted attacks, including Gradient Matching, Bullseye Polytope, and Feature Collision ineffective, with only a slight decrease in the performance. We note that, EPIc is the only effective defense method against state-of-the-art attacks that can efficiently scale to standard deep learning pipelines. Compared to the state-of-the-art [59], EPIc is 6.9x faster, and maintains similarly high test accuracy and low attack success rate.

4.1 Related Work

4.1.1 Targeted Data Poisoning

Attacks on deep networks can be generally divided into triggered and triggerless attacks. Triggered or backdoor attacks augment the training data with a small set of examples that contain a trigger patch and belong to a specific target label. Models trained on the augmented data will misclassify test examples with the same patch. While early backdoor attacks were not clean-label [34, 64, 107, 174], recent backdoor attacks produce poison examples that do not contain a visible trigger [189, 157]. triggerless poisoning attacks add small adversarial perturbations to base images to make their feature representations or gradients match that of the adversarially labeled target [163, 238, 74, 60, 5]. Such poisons are very similar to the base images in the input space, cannot be detected by observers, and do not require modification to targets at inference time. The most prominent poisoning attacks we test our defense against are:

Feature Collision (FC) crafts poisons by adding small perturbations to base examples so that their feature representations collide with that of the target [163].

Bullseye Polytope (BP) is similar to FC, but instead crafts poisons such that the target resides close to the center of their convex hull in feature space [5].

Gradient Matching (GM) produces poisons by approximating this bi-level objective using “gradient alignment", encouraging gradients of the clean-label poisoned data to align with that of the adversarially labeled target [60]. This attack is shown to be effective against data augmentation and differential privacy.

Sleeper Agent (SA) is a hidden-trigger backdoor attack that also craft poisons based on the “gradient alignment" between patched poisons and targets [174].

4.1.2 Defense Strategies

Commonly used data sanitization defenses work by detecting anomalies that fall outside a spherical radius in the feature space [175], spectrum of the feature covariance matrix [186], or activation space [32]. They may also filter points that are labeled differently from their nearest neighbors in the feature space [144]. Such defense mechanisms rely on the assumption that poisons are far from the clean data points in the input or feature space. Hence, they can be easily broken by stronger data poisoning attacks that place poisoned points near one another, or by optimization methods that craft poisons to evade detection [91, 163, 157].

Robust training methods rely on strong data augmentation [22], apply randomized smoothing [198], use an ensemble of models for prediction [97], or bound gradient magnitudes and minimize differences in orientation [70]. Such methods often incur a significant performance penalty [80], and can even be adaptively attacked by modifying gradient signals during poison crafting [194]. Other identify backdoor attacks early in training and revert their effect by gradient ascent [100], use adversarial training [115, 179], or create poisons during training and inject them into training batches [59].

Existing defense methods either drastically degrade the model's performance [59], only protect the model against certain types of poisoning attack [91, 186], are prohibitive for larger datasets [59], or do not provide any theoretical guarantee for the performance of the model [198, 97, 1, 59]. On the other hand, our method is fast and scalable, and successfully eliminates various poisoning attacks while allowing the model to learn effectively from clean examples with rigorous theoretical guarantees.

4.2 Robust Training against Data Poisoning

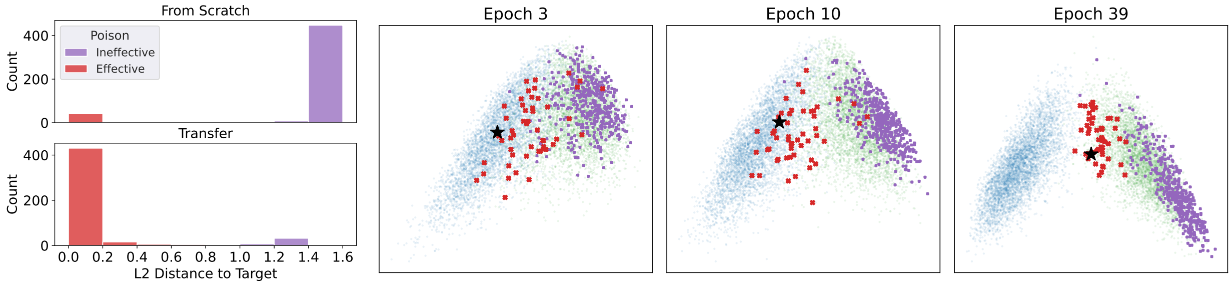
Let $\mathcal{D}_c = \{(x_i, y_i)\}_{i=1}^n$ be the set of all clean training data, where $x_i \in \mathbb{R}^m$. Targeted data poisoning attacks aim to change the prediction of a target image x_t in the test set to an adversarial label y_{adv}, by modifying a fraction (usually less than 1%) of data points in the training data within an l_∞-norm ϵ-bound. We denote by $V = \{1, \cdots, n\}$, and $V_p \subset V$ the index set of the entire training data and poisoned data points, respectively. For small ϵ, this constraint enforces the perturbed images to look visually similar to the original example. Such attacks remain visually invisible to human observers and are called clean-label attacks. Targeted clean-label data poisoning attacks can be formulated as the following bi-level optimization problem:

$$\min_{\delta \in \mathcal{C}} \mathcal{L}(x_t, y_{\text{adv}}, \theta(\delta)) \quad s.t. \qquad \theta(\delta) = \arg\min_\theta \sum_{i \in V} \mathcal{L}(x_i + \delta_i, y_i, \theta), \tag{4.1}$$

where $\mathcal{C} = \{\delta \in \mathbb{R}^{n \times m} \colon \|\delta\|_\infty \leq \epsilon, \delta_i = 0 \ \forall i \notin V_p\}$ is the constraint set determining the set of valid poisons. Intuitively, the perturbations change the parameters θ of the network such that minimizing the training loss on RHS of Eq. (4.1) also minimizes the adversarial loss on LHS of Eq. (4.1).

We assume that the network is trained by minimizing the training loss $\mathcal{L}(\theta) = \sum_{i \in V} \mathcal{L}(x_i + \delta_i, y_i, \theta)$ over the entire set of clean and poisoned training examples $i \in V, \delta_i = 0 \ \forall i \notin V_p$. Applying gradient descent with learning rate η to minimize the training loss $\mathcal{L}(\theta)$, iteration τ takes the form:

$$\theta_{\tau+1} = \theta_\tau - \eta \nabla \mathcal{L}(\theta_\tau). \tag{4.2}$$



(a) Poison distribution (b) Training from scratch for 40 epochs with GM poisons

Figure 4.1: 500 effective (red) and ineffective (purple) poisons crafted by GM and BP in from-scratch and transfer learning scenarios on CIFAR10. (a) Number of effective vs. ineffective poisons and their distance to the target in the (last layer) gradient space of a clean model; (b) Embeddings of effective (red) and ineffective (purple) poisons, and clean examples of the target (blue) and poison (green) class, projected on the first 2 principal components. Effective poisons are not examples with the lowest confidence or highest loss.

Attack and defense assumptions. We consider a worst-case scenario, where the attacker has knowledge of the defender's training procedure (e.g. learning rate, optimization algorithm), architecture, and defense strategy, but cannot influence training, initialization, or mini-batch sampling. In transfer learning where the defender uses a pre-trained model and only trains the last layer, we assume the parameters of the pre-trained model are known to the attacker. However, the defender is not aware of the target example or the specific patch chosen by the attacker. We also assume that the defender does not have access to additional clean data points.

4.2.1 Motivation

For a targeted poisoning attack to be successful, the target needs to be misclassified as the adversarial class y_{adv}. Effectively, the poisons need to pull the representation of the target toward the poison class. To do so, they need to mimic the gradient of the adversarially labeled target. Formally,

$$\nabla \mathcal{L}(x_t, y_{\text{adv}}, \theta) \approx \frac{1}{|V_p|} \sum_{i \in V_p} \nabla \mathcal{L}(x_i + \delta_i, y_i, \theta) \tag{4.3}$$

needs to hold for any θ encountered during training.

This is the motivation behind the poison generation in the end to end training scenario. In particular, Gradient Matching [60] and Sleeper Agent [174] explicitly minimize the alignment (cosine similarity) between poison and target gradient as in Eq. (4.3), using a clean pre-trained model. Although the poisons are generated using a pre-trained clean model, [59] empirically showed that the alignment between the gradient of adversarial and training loss remains large during the training. MetaPoison [74] uses a number of partially-trained models to generate poisons that minimize the adversarial loss at different stages during the training. Bullseye Polytope [5] maximizes the similarity between representations of the poisons and target. In doing so, it implicitly minimizes the alignment between poison and target gradients w.r.t. the penultimate layer, which captures most of the gradient norm variation [87].

In the transfer learning scenario, the poisons are crafted to have a similar representation to that of the target. Here, a linear layer is trained on the poisoned data using the representations obtained from a pre-trained clean model. The gradient of the linear model is proportional to the representations learned by the pre-trained model. Therefore, by maximizing the similarity between the representations of the poisons and the adversarially labeled target, the attack indeed increases the alignment between their gradients.

Crucially, the better the poisons can surround the target in the gradient space, the more effective the attack becomes. This is demonstrated by the superior success rate of Bullseye Polytope [5] and Convex Polytope [238], compared to that of Feature Collision [163]. While Feature Collision only optimizes the poisons to have a similar representation to that of the adversarially labeled target, Convex Polytope moves poisons until the target is inside their convex hull, and Bullseye Polytope makes further refinements to move the target away from the polytope boundary.

4.2.2 Not all the poisons are created equal

To successfully prevent poisoning attacks, we make the following key observation: Not all the poisoned examples are responsible for the success of the attack. We define *effective poisons*

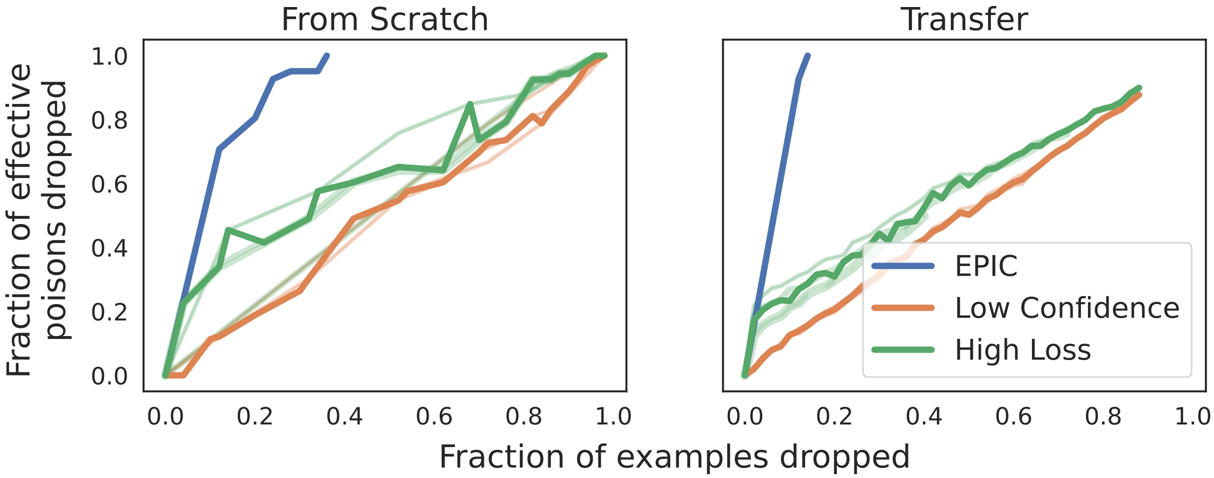
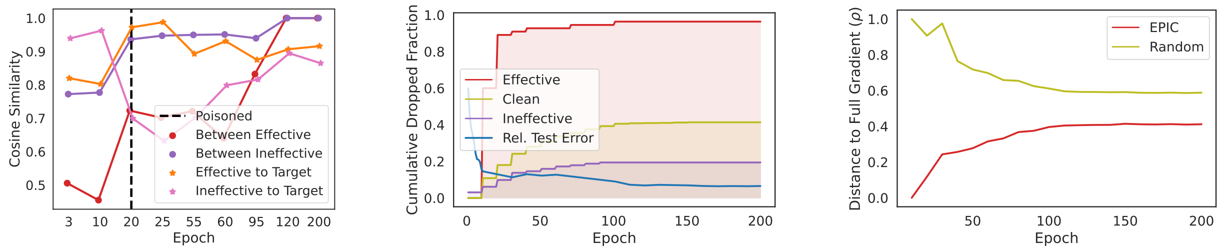


Figure 4.2: Fraction of effective poisons dropped vs fraction of all examples dropped during training on CIFAR10 poisoned with GM, for our method (EPIC) vs lowest-confidence and highest-loss with thresholds .25,.5/1,2 shown by transparent colors, and their average shown in opaque. Left: from scratch. Right: transfer learning.

as examples that make the attack successful. That is, if the model is trained with effective poisons, the attack will be successful even if all the other poisons are removed. In contrast, if the effective poisons are eliminated, the remaining (ineffective) poisons cannot make the attack successful. Fig. 4.1a shows 500 effective and ineffective poisons generated by Gradient Matching (GM) and Bulleyes Polytope (BP) in the training from scratch and transfer learning scenarios. We tried different combinations of the poisons and identified the smallest subset of poisons that is responsible for the success of the attack. We observe that indeed not all poisons are effective. While for from-scratch training only 8% of the poisons are effective, for transfer learning around 90% of the poisons are effective.

We explain the above observation as follows: not all the randomly selected examples can be modified by bounded perturbations to have a gradient that closely matches that of the target. When training from scratch, attacks can only craft a handful of effective poisons as the poisons need to match the very high-dimensional gradient of the target with bounded perturbations. On the other hand, during transfer learning, poisons are optimized to match the much lower-dimensional gradient of the target. Hence, attacks can craft a much larger number of effective poisons.



(a) Cosine similarity between effective/ineffective poisons and the target.

(b) Cumulative fraction of (in)effective poisons & cleans dropped by EPIC vs test error.

(c) Gradient difference of examples not dropped with EPIC vs random, with full data.

Figure 4.3: Training with EPIC on CIFAR10 poisoned with GM. (a) The similarity between effective poisons' gradients to each other becomes small (they get isolated) after the warmup period, (b) EPIC effectively eliminates effective poisons while dropping a smaller fraction of clean examples, (c) EPIC preserves main gradient components, hence remaining examples have a closer gradient to that of the full data, compared to random subsets of the same size. Thus, EPIC preserves the training dynamics.

4.2.3 Effective poisons are not examples with highest loss or lowest confidence

It is important to note that effective poisons are not the data points around the decision boundary for which the model is not confident, or outliers that have a higher loss than other data points in their class. Fig. 4.1b shows the embedding of clean and poisoned examples of the poison and target class during training from scratch. We see that effective poisons can be within the poison or target class, or at the boundary of the classes, at different training iterations. Fig. 4.2 shows the fraction of effective poisons eliminated when we drop examples with the highest loss or lowest confidence with various thresholds during the training. We see that dropping lowest-confidence or highest-loss examples during the training indeed removes a larger number of *clean* data points, and cannot successfully eliminate the effective poisons.

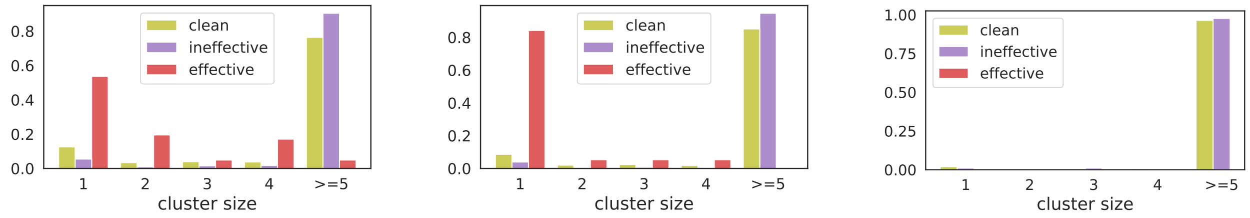
4.2.4 Effective poisons become isolated in gradient space

Attacks exploit the non-convex nature of the neural network loss to optimize poisons that match the target gradient. For ineffective poisons, attacks cannot successfully modify the base example with bounded perturbations to match the target gradient. This is the case where loss is relatively smooth in a ball of radius ϵ around the base. Thus, for ineffective poisons, attacks can only increase the alignment between the gradients of ineffective poisons with that of the target to some extent. In doing so, the similarity between ineffective poisons' gradients becomes larger. Hence, they form larger gradient clusters in the poison class, as shown by Fig. 4.1b.

On the other hand, effective poisons can be modified under bounded perturbations to match the target gradient. This is the case where there are sharp regions in a ball of radius ϵ around a base example. Here, the base can be perturbed and taken to such sharp regions, and its gradient can be further optimized there to match the target gradient. During the training on the poisoned data, the gradients of effective poisons move far away from their class and get close to the target. But, they each have a *different trajectory* (starting from different base examples) for interpolating between their base and the target gradients. These trajectories are neither similar to each other (as they start from different bases) nor similar to other examples in the base class (as they end up matching the target's gradient in another class). Fig. 4.3a shows that while the similarity between gradients of effective poisons and target increases during the training, the gradient of effective poisons is very different from each other after a few epochs of training, and before the model gets poisoned. Hence, effective poisons' gradients become isolated in the gradient space, early in training. Such isolated points in low-density gradient regions can be best identified by proximity-based methods, such as k-medoids, as we discuss next.

4.2.5 Eliminating the effective poisons

To prevent data poisoning while maintaining the generalization performance of the network, we aim at identifying and removing the effective poisons. To do so, our key idea is to drop data points that have a different gradient compared to other examples in their class, i.e., are isolated in the gradient space during training. As we discuss next, dropping such points effectively eliminates the majority of poisoning attacks with only a slight impact on the gradient of the full training loss. By preserving the important gradient components, we guarantee similar training dynamics and convergence to a close neighborhood of the solution obtained by training on full data. To find the effective poisons that do not have a similar

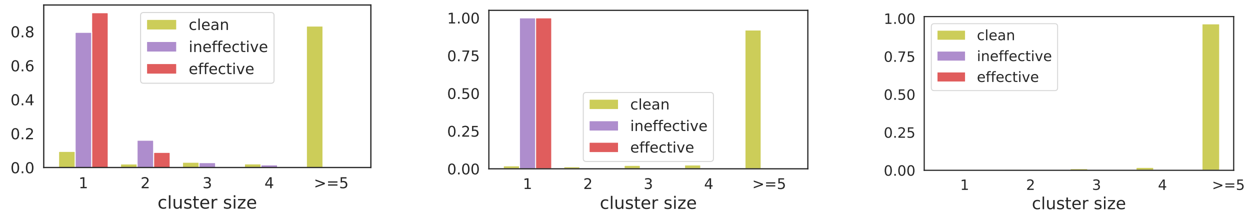


(a) Epoch 10. Most of the effective poisons are isolated as clusters of size 1.

(b) Epoch 20. Most of remaining effective poisons become isolated.

(c) Epoch 80. Clean examples form larger gradient clusters during the training.

Figure 4.4: Fraction of clean vs Gradient Matching poisons in gradient clusters of different sizes, during from-scratch learning with EPIC for 200 epochs. Effective poisons become isolated during training and can be iteratively eliminated by EPIC.



(a) Epoch 1. Most of the effective poisons are isolated as clusters of size 1.

(b) Epoch 2. Remaining effective poisons become isolated as clusters of size 1.

(c) Epoch 20. Clean examples form larger gradient clusters during the training.

Figure 4.5: Fraction of clean vs Bullseye Polytope poisons in gradient clusters of different sizes, during transfer learning with EPIC for 40 epochs. Effective poisons become isolated during training and can be iteratively eliminated by EPIC.

gradient to the other data points in their class, we train the model for a few epochs (warmup). Then, we iteratively find and drop the isolated points in low-density gradient regions. To do so, we find the *medoids*—the most centrally located data points—of each class, in the gradient space. Then, we assign every data point to its closest medoid, and drop the medoids to which no other data point is assigned. In our experiments, we show that selecting small subsets (10%-30% of data) of medoids at every iteration can successfully prevent various types of data poisoning attacks. Fig. 4.4, 4.5 show that during training from scratch or transfer learning, effective poisons are isolated medoids of the gradient space. Hence, our strategy successfully identifies the majority of the effective poisons in both from scratch and transfer learning settings, while only dropping a small number of clean examples (Fig. 4.2, 4.3b).

The set of medoids of a class minimizes the average gradient dissimilarity to all the other

Algorithm 3 Effective Poison Identification (EPIC)

```
Input: Training data indexed by $V$, submodular facility location function $F$, loss function $\mathcal{L}(\cdot)$, warmup iterations $K$, poison drop interval $T$, number of medoids $k$.
Output: Subset $S \subseteq V$
Train the network for $K$ epochs on $V$.
for every $T$ epochs do
    for examples $V_c$ in class $c \in [C]$ do
        Initialize $S \leftarrow \emptyset, Z \leftarrow \emptyset$
        while $|S| < k/C$ do
            $j \in \arg\max_{e \in V_c \setminus S} F(e|S)$
            $S = S \cup \{j\}$
        end while
        for $j = 1$ to $|S|$ do
            $\gamma_j = \sum_{i \in V_c} \mathbb{I}[j = \text{argmin}_{s \in S} ||\nabla_f^L \mathcal{L}_i(\theta) - \nabla_f^L \mathcal{L}_s(\theta)||]$
            if $\gamma_i == 1$ then
                $Z = Z \cup \{j\}$
            end if
        end for
        $V = V \setminus Z$
    end for
    Train on $V$ for $T$ epochs.
end for
```

data points in the class. For a specific value of k, the set of k-medoids can be found as:

$$S_\tau^* \in \arg\min_{\substack{S \subseteq V \\ |S| \leq k}} \sum_{i \in V} \min_{j \in S} \|\nabla \mathcal{L}_i(\theta_\tau) - \nabla \mathcal{L}_j(\theta_\tau)\|_2, \tag{4.4}$$

where $\mathcal{L}_i(\theta) = \mathcal{L}(x_i, y_i, \theta)$ is the loss associated with (potentially poisoned) training example $i \in V$. The minimization problem (4.4) is NP-hard. However, it can be turned into maximizing a submodular[1] facility location objective:

$$\begin{aligned} S_\tau^* \in \arg\min_{S \subseteq V} |S|, \quad s.t. \\ F(S) = \sum_{i \in V} \max_{j \in S} c_0 - \|\nabla \mathcal{L}_i(\theta_\tau) - \nabla \mathcal{L}_j(\theta_\tau)\|_2, \end{aligned} \tag{4.5}$$

where c_0 is a constant satisfying $c_0 \geq \|\nabla \mathcal{L}_i(\theta_\tau) - \nabla \mathcal{L}_j(\theta_\tau)\|_2$, for all $i, j \in V$. For maximizing

[1] A set function $F : 2^V \rightarrow \mathbb{R}^+$ is submodular if $F(S \cup \{e\}) - F(S) \geq F(T \cup \{e\}) - F(T)$, for any $S \subseteq T \subseteq V$ and $e \in V \setminus T$. F is *monotone* if $F(e|S) \geq 0$ for any $e \in V \setminus S$ and $S \subseteq V$.

a monotone submodular function, the greedy algorithm provides a $(1 - 1/e)$ approximation guarantee [203]. The greedy algorithm starts with the empty set $S_0 = \emptyset$, and at each iteration t, it chooses an element $e \in V$ that maximizes the marginal utility $F(e|S_t) = F(S_t \cup \{e\}) - F(S_t)$. Formally, $S_t = S_{t-1} \cup \{\arg\max_{e \in V} F(e|S_{t-1})\}$. The computational complexity of the greedy algorithm is $\mathcal{O}(nk)$. However, its complexity can be reduced to $\mathcal{O}(|V|)$ using stochastic methods [125], and can be further improved using lazy evaluation [123] and distributed implementations [124].

During the training, the gradients of data points change at every iteration. To identify the effective poisons, we need to update the gradient medoids iteratively. The gradient vectors can be very high-dimensional, in particular when training from scratch. To efficiently solve Eq. (4.5), we rely on a recent result showing that the variation of the gradient norms is mostly captured by the gradient of the loss w.r.t. the input to the last layer [87]. Hence, upper-bound on the normalized difference between pairwise gradient dissimilarities can be efficiently calculated:

$$\|\nabla \mathcal{L}_i(\theta_\tau) - \nabla \mathcal{L}_j(\theta_\tau)\|_2 \leq \mathcal{O}\big(\|\nabla_f^L \mathcal{L}_i(\theta_\tau) - \nabla_f^L \mathcal{L}_j(\theta_\tau)\|_2\big)$$

where $\nabla_f^L \mathcal{L}_i$ is gradient of the loss function $\mathcal{L}$ w.r.t. the input to the last layer L for data point i. The above upper bound is marginally more expensive to calculate than loss. Hence, upper bounds on the gradient dissimilarities can be efficiently calculated. Alg. 1 illustrates the pseudocode.

Iteratively eliminating the isolated medoids during the training allows us to successfully prevent various types of attacks. At the same time, as EPIc drops scattered gradient outliers and doesn't skew larger (main) gradient clusters, it only introduces a small limited error (ρ) on the full training gradient. Fig. 4.3c shows that the gradient of the remaining training examples during training with EPIc is much closer to the full training gradient, compared to that of random subsets of the same size. Theorem 4.2.1 leverages this idea to upper-bound the difference between the loss of the model trained with EPIc and the model trained on the full data, at every step of training. This ensures similar training (loss) dynamics to that of training on the full data, and allows the network to obtain a similar generalization performance.

Table 4.1: Average attack success rate and validation accuracy for EPIC against various data poisoning attacks (200-epoch pipeline).

Attack	Senario	Undefended		Defended	
		Att Succ.↑	Test Acc.↑	Att Succ.↓	Test Acc.↑
Gradient Matching	from-scratch	45%	94.95%	1%	90.26%
Sleeper Agent (backdoor)	from-scratch	78.54%	94.42%	11.55%	88.28%
Bullseye Polytope	transfer	86%	94.69%	1%	94.80%
Feature Collision	transfer	40%	94.68%	0%	94.81%
Bullseye Polytope	finetune	80%	92.24%	0%	92.38%

Theorem 4.2.1. *Assume that the loss function $\mathcal{L}(\theta)$ is μ-PL* on a set Θ, i.e., $\frac{1}{2}\|\nabla\mathcal{L}(\theta)\|^2 \geq \mu\mathcal{L}(\theta), \forall\theta \in \Theta$. Assume ρ is the maximum change in the gradient norm due to dropping points. Then, applying gradient descent with a constant learning rate η has similar training dynamics to that of training on the full data. I.e.,*

$$\mathcal{L}(\theta_t) \leq (1-\eta\mu)^t\mathcal{L}(\theta_0) - \frac{1}{2\mu}(\rho^2 - 2\rho\nabla_{\max}). \tag{4.6}$$

The proof can be found in the Appendix.

Compared to existing defense strategies, our method does not require a pre-trained clean model, is not attack-specific, can be applied very efficiently during the training, and provides a quality guarantee for the performance of the trained model.

4.2.6 Adaptive attacks

Adaptive attacks can generate more powerful poisons by taking into account the knowledge of the particular defense mechanisms in place. For example, Gradient Matching [60] and Sleeper Agent [174] include augmented data points that are transformed with e.g. crop and flip in addition to the original ones during poison crafting in Eq. (4.3). In doing so, the attack can successfully poison the model even when data augmentations like crops and flips are applied to the learning pipeline. For adaptive attacks to be successful in presence of EPIc, they need to generate clusters of effective poisons. To do so, the attacker may craft poisons with similar gradient trajectories during the training, or optimize the choice of base examples that result in clustered poison gradients. However, crafting poisons with similar gradient trajectories during the training makes the poison optimization prohibitive and may result in less effective attacks. While selecting similar base images does not lead to clustered effective poisons due to non-convex nature of loss, optimizing the choice of base examples worth further investigation in future work.

Next, we show that our method achieves superior performance compared to existing defense techniques.

4.3 Experiments

4.3.1 Against Data Poisoning Attacks

We evaluate the effectiveness of defense methods against data poisoning attacks, during *from-scratch training*, *transfer learning* and *fine-tuning*. For our evaluation, we use the standardized data poisoning benchmark [161], with 200 training epochs, starting learning rate of 0.1, and a decaying factor of 10 at epochs 100, 150. As several defense methods are prohibitive to be applied to a standard learning pipeline with 200 epochs, we also consider a *proxy* setup used by [59] which trains for only 40 epochs, with a starting learning rate of 0.1 and decaying factor of 10 at epochs 25, 35.

4.3.1.1 From-Scratch Training

We model the from-scratch training experiments based on the benchmark setting [161]. For our attack model, we select 1% of the training examples as poisons, which are perturbed within the l_∞ ball of radius $\epsilon = 8/255$. The defender initializes a model based on a random seed and trains on the poisoned dataset using SGD. To maximize reproducibility, we only use publicly available poisoned datasets generated by authors of the attacks.

Table 4.2: Avg. Poison Success versus validation accuracy for various defenses against the gradient matching attack of [60] in the from-scratch setting. The proposed Robust Training Against Data Poisoning is listed as EPIC.

Epoch	Defense	Attack Succ.↓	Test Acc.↑	Time(hr:min)
40	None	25%	92.48%	00:15
40	DeepKNN [144]	21%	91.86%	02:25
40	Spectral Signatures [186]	17%	90.13%	00:40
40	Activation Clustering [32]	9%	84.20%	00:31
40	Diff. Priv. SGD [70]	2%	70.34%	00:16
40	Adv. Poisoning-0.25 [59]	4%	91.48%	01:53
40	Adv. Poisoning-0.5 [59]	1%	90.67%	02:02
40	Adv. Poisoning-0.75 [59]	0%	87.97%	02:26
40	EPIC-0.1 (Proposed)	2.7%±0.6%	90.92%±0.26%	00:22
40	EPIC-0.2 (Proposed)	1.3%±0.6%	88.95%±0.08%	00:19
40	EPIC-0.3 (Proposed)	1.0%±0.0%	87.03%±0.11%	00:17
200	None	45%	94.95%	01:18
200	Spectral Signatures [186]	10%	92.99%	03:22
200	Activation Clustering [32]	11%	90.88%	02:33
200	Diff. Priv. SGD [70]	2%	80.71%	01:23
200	EPIC-0.1 (Proposed)	2.3%±0.6%	92.50%±0.03%	01:50
200	EPIC-0.2 (Proposed)	1.0%±1.0%	89.71%±0.06%	01:35
200	EPIC-0.3 (Proposed)	0.7%±0.6%	87.05%±0.05%	01:28

Unless otherwise specified, we augment training images with random horizontal flip followed by random cropping, and per-channel normalization. For our proposed defense, we run EPIC with $T=2$ in a 40-epoch training pipeline, or $T=10$ in a 200-epoch training pipeline.

Warmup The more medoids we select for each class, the longer warmup period (K) we need for EPIC. In the experiments, we set $K = 10$ for EPIC-0.1, $K = 20$ for EPIC-0.2 and $K = 30$ for EPIC-0.3.

Gradient Matching (GM) GM is currently the state-of-the-art among data poisoning attacks for from-scratch training [161]. [60] shows it significantly outperform the other effective attack, MetaPoison [74]. We test 100 datasets provided by the authors . The

Table 4.3: Defending against the BP attack on TinyImageNet while training from scratch. Our method (EPIc) can train more accurate models than the SOTA defense (AP) without increasing the success rate of poisoning attacks, and is more scalable.

Defense	Attack Succ.↓	Test Acc.↑	Time↓
None	40%	61.80%	1hr
AP-0.5	0%	53.54%	7hrs
EPIc-0.2	0%	57.50%	1hr

datasets were generated based on the 100 preset benchmark settings, each with 500 specific bases and a target image. We follow the training hyperparameters specified by the benchmark to train ResNet-18 from scratch with 128 examples per mini-batch. Table 4.1 shows that the average attack success rate of GM on these 100 datasets is 45% and the average test accuracy is 94.95%. We see that our proposed defense, EPIc, is able to successfully drop the average attack success rate to only 1% while keeping the test accuracy above 90%.

Bullseye Polytope (BP) [161] shows the superiority of BP attack in training VGG models [167] from scratch on TinyImageNet, a subset of the ILSVRC2012 classification dataset [51]. We tested the first 10 example datasets provided by the benchmark, and observed an attack success rate of 40%. Training with EPIc drops the attack success rate significantly to 0%, as shown in Table 5.4.

Sleeper Agent (SA) SA is the only backdoor attack that can achieve a higher than single-digit success rate on CIFAR-10 in the from-scratch setting. We generated 20 poisoned datasets with SA ($\epsilon = 16$) using the source-target class pairs in the first 20 CIFAR-10 benchmark settings. For backdoor attacks, we evaluate the attack success rate over 1000 patched test images. The average attack success rate is 78.54% without defenses and 11.55% with EPIc.

4.3.1.2 Transfer Learning

Here, we use the 40-epoch pipeline of [59] to evaluate defense methods. The same pretrained model is used for generating the attack and for transfer learning onto the defender. Similar to the from-scratch setting, the attacker can modify 1% of 50000 training examples in CIFAR10 with $\epsilon = 8$. The linear layer (classifier) of the pretrained model is then re-initialized and trained with the poisoned dataset with all the other layers (feature extractor) fixed during the training. This white-box setup allows attacks to produce stronger poisons. Here, we apply EPIC with $T = 1$.

Bullseye Polytope (BP) According to [161], Bullseye Polytope attack [5] has the highest average attack success rate in the white-box setting. When evaluated on all 100 benchmark setups, BP succeeded in 86 of them. Table 4.1 shows that EPIC could successfully drop the attack success rate to only 1% while even increasing the test accuracy of the model.

Feature Collision (FC) As imposing the l_∞ constraint $\epsilon = 8$ will greatly reduce the power of Feature Collison attack [161], we keep the l_2 regularization term in their original optimization objective to impose a soft rather than hard constraint on the l_∞ perturbation. We generate 20 poisoned datasets using the first 20 benchmark setups (indexed from 0 to 19). With the default seed used by the benchmark, the attack success rate of these 20 datasets generated by FC is 40% before and 0% after we apply our EPIC defense, as shown in Table 4.1.

4.3.1.3 Finetuning

We also consider the finetuning scenario in which the classifier is re-initialized and the feature extractor is not fixed during the training. We follow the same setup in [59], test 20 datasets poisoned with BP, and report the result in Table 4.1. Again, EPIC successfully prevents all attacks in this scenario without decreasing the test accuracy.

4.3.2 Comparison to SOTA Defenses against GM

Table 4.2 compares the effectiveness of our model with existing defense methods against the state-of-the-art GM attack, in both 200-epoch and 40-epoch training scenarios. We see that EPIC can successfully drop the success rate of GM while allowing the model to achieve superior performance. We note that, unlike existing defense methods, our method is easily scalable to standard deep learning pipelines.

Scalability As many defense methods [59, 144] are prohibitive when applied to the standard 200-epoch pipeline under time or space constraints, they are evaluated using a 40-epoch pipeline. However, as Table 4.2 shows, training a model on the same poisoned datasets for more epochs increases attack success rate. Therefore, defenses that are successful within 40 epochs are not guaranteed to have the same effectiveness when models are trained for longer. On the contrary, our proposed defense requires nearly no extra time compared to normal training. Time spent on running EPIC every few epochs is usually well compensated by training time saved every epoch on the examples we drop. Table 4.2 includes the time for each defense on CIFAR10 poisoned with GM, and Table 5.4 compares EPIC with AP on TinyImagenet poisoned with BP. We report the wall-clock time of training a model with each defense on a single NVIDIA A40 GPU with 4 workers. We see that EPIC effectively reduces various attacks' success rates while having substantially faster run time.

Strength of Defense Due to computational constraints and the scalability problem mentioned above, we only scale the two general adversarial training methods, Adversarial Training [115] and DP-SGD [70] to the standard 200-epoch training pipeline. According to Table 4.2, 5.4 and Fig. 4.6, our method provides the best trade-offs between the defended attack success rate and the overall test accuracy. Adversarial Poisoning [59] can give equally good trade-offs but requires 6x training time. Other defenses either cannot guarantee a low attack success rate or have a high computation cost.

Table 4.4: Comparison of avg. poison accuracy, validation accuracy and time against the strongest attack GM [60] with $\epsilon = 16$ in the from-scratch setting for 40 epochs. Our proposed defense is listed as EPIc.

Defense	Attack Succ.↓	Test Acc.↑
None	90%	92.01%
AP-0.25	35%	91.21%
AP-0.5	10%	90.58%
AP-0.75	0%	87.97%
EPIc-0.1	10%	91.15%
EPIc-0.2	0%	89.07%

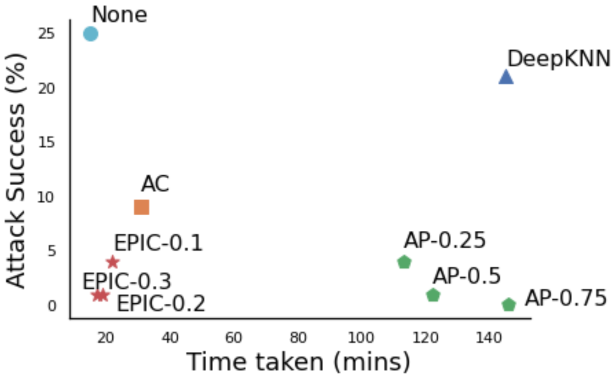


Figure 4.6: Attack success rate vs. running time of different defenses, for GM attack on CIFAR-10 with the 40 Epochs pipeline.

4.3.3 Comparison under Larger Perturbations

Attacks usually have higher success rates when allowed to perturb the base images within a larger ϵ constraint [161]. We generate 20 poisoned datasets with a larger $\epsilon = 16$ with GM. We use the default 20 seeds used in [60] to sample 500 base and 1 target images from CIFAR10. Table 4.4 shows that for larger ϵ, EPIc achieves a superior performance compared to the strongest baseline, adversarial poisoning.

4.4 Conclusion

We proposed an efficient defense mechanism against various data poisoning attacks. We showed that under bounded perturbations, only a small number of poisons can be optimized to have a gradient that is close enough to that of the target and make the attack successful. Such examples move away from their original class and get isolated in the gradient space. Consequently, we showed that training on large gradient clusters of each class successfully eliminates the effective poisons, and guarantees similar training dynamics to that of training on the full data. Our experiments showed that our method significantly decreases the success rate of the state-of-the-art targeted attacks, including Gradient Matching, Bullseye Polytope. We note that our method is the only effective defense against strong poisoning attacks, which easily scales to standard deep learning pipelines.

Part III

Data-efficient and Robust Training against Spurious Correlations

CHAPTER 5

SPARE: Identifying Spurious Correlations Early in Training

The *simplicity bias* of gradient-based training algorithms towards learning simpler solutions has been suggested as a key factor for the superior generalization performance of overparameterized neural networks [69, 71, 131, 136, 146, 164]. At the same time, it is conjectured to make neural networks vulnerable to learning *spurious* correlations frequently found in real-world datasets [155, 172]. Neural networks trained with gradient-based methods can exclusively rely on simple spurious features that exist in majority of examples in a class but are not predictive of the class in general (e.g., image background), and remain invariant to the predictive but more complex core features [164]. This results in learning non-robust solutions that do not generalize well under seemingly benign distribution shifts, or on minority groups of the original data distribution that do not contain the spurious features [164, 181].

To alleviate the effect of spurious biases in absence of group labels, existing methods partition examples in each class into a majority and a minority group. This is achieved by training the model with gradient descent and identifying the minority group as examples that are misclassified [106], have a high loss [132], or sensitive representations [43, 172], or form small representation clusters at the end of training [6, 229]. Then, a robust model is trained while upweighting [155] or upsampling [106] the minority group, or via supervised contrastive learning [229], to alleviate the effect of the spurious feature of the majority group. This can be problematic for many real-world datasets. First, they cannot distinguish between multiple minority groups. If minority groups are imbalanced, upweighting or upsampling them to the same extent can introduce new spurious biases. Second, they tend to find *unusual* examples in the minority group. While unusual examples in curated spurious benchmark datasets are only those without the spurious feature, more realistic datasets such as ImageNet, contain outliers and noisy examples, which gather in the minority group. Upweighting or upsampling such examples harm the worst-group and total accuracy. Finally, state-of-the-art methods may increase the training time by orders of magnitude [106, 229], and become prohibitive for even medium-sized datasets.

In this work, we show that the simplicity bias of gradient descent that results in learning the spurious biases can be leveraged to provably separate majority and (multiple) minority groups, *early in training*. In particular, we analyze a two-layer fully connected neural network and identify two phases early in training. First, in the initial training iterations, the contribution of the spurious feature in a majority group to the model's output increases linearly by the amount of spurious correlation. Afterwards, if the noise-to-signal ratio of a spurious feature is smaller than that of the core feature, the model's output on the majority of examples in the class is almost exclusively determined by the spurious feature, and will be invariant to the core features. We show that the model's output *provably* separates majority and (multiple) minority groups early in training. Based on the above theoretical insights, we propose a method SPARE (SePArate early and REsample), which clusters the model's output early in training to find a number of groups, determined by the silhouette score. Then, it applies importance sampling to make the groups relatively balanced. In doing so, it effectively alleviates the effect of spurious bias without increasing the training time.

Through extensive experiments, we confirm that SPARE can achieve up to 5.6% higher worst-group accuracy compared to the state-of-the-art baselines that infer the group information on CMNIST [8], Waterbirds [155], and CelebA [110], while being up to 12x faster. Notably, SPARE achieves a comparable or even better performance compared to methods that fully or partially know the group information at training time, and is highly effective in the presence of multiple minority groups and extreme group imbalance. We also apply SPARE to discover and mitigate spurious correlations in a more realistic setting, namely in Restricted ImageNet [178]. Our results confirm the effectiveness of SPARE in finding and alleviating the spurious bias, instead of upweighting the unusual examples.

5.1 Related Work

Mitigating spurious bias. If group labels are known at training time, class balancing techniques [66, 45], or importance weighting [166, 27] are applied to improve the performance on smaller groups. Alternatively, one can directly minimize the worst group-level error among these groups via group robust optimization (GDRO) [155], where training is focused on examples from higher-loss groups.

If group labels are not available, existing methods aim to first infer the group information, and utilize them to robustly train the model for the second time. GEORGE [172] infers group information by clustering ERM (Expected Risk Maximization) representations and trains the second model with GDRO. LfF [132] trains two models simultaneously and upweight examples that have a high loss according to the first model while training the second model. JTT [106] and CNC [229] identify the minority group as those misclassified by the initial ERM model and upsample the minority groups. JTT trains the second robust model using ERM, and CNC applies contrastive learning to pull misclassified examples towards their class. EIIL [43] and PGI [6] rely on a reference ERM model to split training data into majority and minority groups by finding an assignment that maximizes the Invariant Risk Minimization (IRM) objective [11], i.e., the variance of the model on the two groups. Then, EIIL trains the second robust model with GDRO, and PGI minimizes the KL divergence of softmaxed logits for same-class samples across groups. CIM [178] learns input-space transformations of the data to ensure that the transformation preserves task-relevant information. Finally, if a smaller group-labeled data is available, SSA [133] applies semi-supervised learning with extra group-labeled data to infer the training group labels and then uses GDRO to train a robust model. DFR [90] first trains the model with ERM, and then retrains the last layer on the group-balanced data.

State-of-the-art methods either require extra group-labeled data [133, 90], or may fail in presence of multiple imbalanced minority groups [133, 178, 132], and noisy examples [43, 110]. Some increases the training time by orders of magnitude [106, 133, 229]. In contrast, SPARE effectively separates the groups early in training and provide superior performance without increasing the training time.

Simplicity Bias. A recent body of work revealed the simplicity bias of (stochastic) gradient methods towards learning linear functions early in training, followed by functions of increasing complexity in later phases [69, 71, 131, 136, 146, 164]. This phenomenon is empirically observed on various fully connected (FC), convolutional, and sequential networks, such as MobileNetV2 [158], ResNet50 [67], and DenseNet121 [164]. Recently, [71] formally proved that learning dynamics of gradient descent on a two-layer FC neural network can be initially mimicked by a linear model and extended this result to multi-layer FC and convolutional networks. Simplicity bias is suggested as a reason for the good generalization performance of overparameterized neural networks. At the same time, it is *conjectured* to yield models that exclusively rely on the simplest feature and remain invariant to all predictive complex features, even when the simplest feature has less predictive power [164, 181]. However, the exact notion of the simplicity of features and the mechanism by which they are learned remain poorly understood except in certain simplistic settings [130, 164]. Here, we build on [71] and rigorously specify the required conditions and mechanism of learning spurious features by a two-layer FC network.

5.2 Problem Formulation

Let $\mathcal{D}=\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$ be n training data with features $\boldsymbol{x}_i \in \mathbb{R}^d$, and labels $y_i \in \mathcal{C} = \{1,-1\}$.

Features & Groups. We assume every class $c \in \mathcal{C}$ has a *core* feature $\boldsymbol{v}_c$, which is the invariant feature of the class that appears in both the training and test set. Besides, a set of *spurious* features $\boldsymbol{v}_s \in \mathcal{A}$ are shared between classes but may not be present at test time. For example, in the CMNIST dataset containing images of colored hand-written digits, the digit is the core feature, and its color is the spurious feature. Assuming w.l.o.g. that all $\boldsymbol{v}_c, \boldsymbol{v}_s \in \mathbb{R}^d$ are orthogonal vectors, the feature vector of every example $\boldsymbol{x}_i$ in class c can be written as $\boldsymbol{x}_i = \boldsymbol{v}_c + \boldsymbol{v}_s + \boldsymbol{\xi}_i$, where $\boldsymbol{v}_s \in \mathcal{A}$, and each $\boldsymbol{\xi}_i$ is a noise vector drawn i.i.d. from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\xi)$. We assume the noise along each feature is independent, and denoted by σ_c^2, σ_s^2 variance of the noise in the directions of $\boldsymbol{v}_c$, $\boldsymbol{v}_s$, respectively. Training examples can be partitioned into groups $g_{c,s}$ based on the combinations of their core and spurious features $(\boldsymbol{v}_c, \boldsymbol{v}_s)$. If a group

$g_{c,s}$ contains the majority of examples in class c, it is called a majority group. A class may contain multiple minority groups corresponding to different spurious features.

Neural Network & Training. We consider a two-layer FC neural network with m hidden neurons:

$$f(\boldsymbol{x};\boldsymbol{W},\boldsymbol{z}) = \frac{1}{\sqrt{m}}\sum_{r=1}^{m} z_r\phi(\boldsymbol{w}_r^T\boldsymbol{x}/\sqrt{d}) = \frac{1}{\sqrt{m}}\boldsymbol{z}^T\phi(\boldsymbol{W}\boldsymbol{x}/\sqrt{d}), \tag{5.1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the input, $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m]^T \in \mathbb{R}^{m\times d}$ is the weight matrix in the first layer, and $\boldsymbol{z} = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_m]^T \in \mathbb{R}^m$ is the weight vector in the second layer. Here $\phi : \mathbb{R} \to \mathbb{R}$ is a smooth or piece-wise linear activation function (including ReLU, Leaky ReLU, Erf, Tanh, Sigmoid, Softplus, etc.) that acts entry-wise on vectors or matrices. We consider the following ℓ_2 training loss:

$$\mathcal{L}(\boldsymbol{W},\boldsymbol{z}) = \frac{1}{2n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i;\boldsymbol{W},\boldsymbol{z}) - y_i)^2. \tag{5.2}$$

We train the network by applying gradient descent on the loss (5.2) starting from random initialization[1]:

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}_t,\boldsymbol{z}_t), \qquad \boldsymbol{z}_{t+1} = \boldsymbol{z}_t - \eta\nabla_{\boldsymbol{z}}\mathcal{L}(\boldsymbol{W}_t,\boldsymbol{z}_t), \tag{5.3}$$

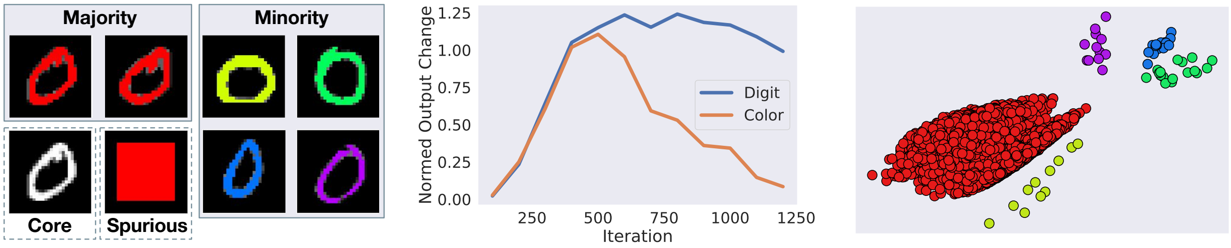
Worst-group error. We quantify the performance of the model based on its highest test error across groups $\mathcal{G} = \{g_{c,s}\}_{c,s}$ in all classes. Formally, *worst-group* test error is defined as:

$$\text{Err}_{wg} = \max_{g\in\mathcal{G}} \mathbb{E}_{(\boldsymbol{x}_i,y_i)\in g}[y_i \neq y_f(\boldsymbol{x}_i;\boldsymbol{W},\boldsymbol{z})], \tag{5.4}$$

where $y_f(\boldsymbol{x}_i;\boldsymbol{W},\boldsymbol{z})$ is the label predicted by the model. In other words, Err_{wg} measures the highest fraction of examples that are incorrectly classified across all groups.

While for simplicity, we consider binary classification with ℓ_2 loss, our analysis generalizes to multi-class classification with CE loss, and other model architectures, as we also confirm

[1] Detailed assumptions on the activations, and initialization can be found in Appendix 9.4.2



(a) Class 0

(b) Change of output when inputting only digit (core) or color (spurious)

(c) PCA visualization of the network outputs on class 0

Figure 5.1: Training LeNet-5 on Colored MNIST containing colored handwritten digits. (a) Each digit is a class; the majority of digits in a class have a particular color, and the remaining digits are in 4 other colors. (b) The network output is almost exclusively indicated by the color of the majority group, early in training. (c) Majority and minority groups are separable based on the network output.

experimentally.

5.3 Investigating How Spurious Features are Learned by Neural Networks

We start by investigating how spurious features are learned during training a two-layer fully-connected neural network. Our analysis reveals two phases in early-time learning. First, in the initial training iterations, the contribution of a spurious feature to the network output increases linearly with the amount of the spurious correlation. Interestingly, if the majority group is sufficiently large, majority and minority groups are separable at this phase by the network output. Second, if the noise-to-signal ratio of the spurious feature of the majority group is smaller than that of the core feature, the network's output on the majority of examples in the class will be almost exclusively determined by the spurious feature and will remain mostly invariant to the core feature. Next, we will discuss the two phases in detail.

5.3.1 Spurious Features are Learned in the Initial Training Iterations

We start by analyzing the effect of spurious features on the learning dynamics of a two-layer FC neural network trained with gradient descent in the initial training iterations. The following theorem shows that if a majority group is sufficiently large, the contribution of the spurious feature of the majority group to the model's output is magnified by the network at every step early in training.

Theorem 5.3.1. *Let $\alpha \in (0, \frac{1}{4})$ be a fixed constant. Suppose the number of training samples n and the network width m satisfy $n \gtrsim d^{1+\alpha}$ and $m \gtrsim d^{1+\alpha}$. Let n_c be the number of examples in class c, and $n_{c,s} = |g_{c,s}|$ be the size of group $g_{c,s}$ with label c and spurious feature $\boldsymbol{v}_s \in \mathcal{A}$. Then, under the setting of Sec. 5.2 there exist a constant $\nu_1 > 0$, such that with high probability, for all $0 \leq t \leq \nu_1 \cdot \sqrt{\frac{d^{1-\alpha}}{\eta}}$, the contribution of the core and spurious features to the network output can be quantified as follows:*

$$f(\boldsymbol{v}_c; \boldsymbol{W}_t, \boldsymbol{z}_t) = \sqrt{\frac{2}{d}}\eta\zeta c\|\boldsymbol{v}_c\|^2 t\left(\frac{n_c}{n} \pm \mathcal{O}(d^{-\Omega(\alpha)})\right), \tag{5.5}$$

$$f(\boldsymbol{v}_s; \boldsymbol{W}_t, \boldsymbol{z}_t) = \sqrt{\frac{2}{d}}\eta\zeta c\|\boldsymbol{v}_s\|^2 t\left(\frac{n_{c,s} - n_{c',s}}{n} \pm \mathcal{O}(d^{-\Omega(\alpha)})\right), \tag{5.6}$$

where $c' = \mathcal{C} \backslash c$, and ζ is the expected gradient of activation functions at random initialization.

The proof can be found in Appendix 9.4.3.2. Note that the width requirement in Theorem 5.3.1 is very mild as it only requires to be larger than $d^{1+\alpha}$ for some small constant α, but can be much smaller than the number of samples. The proof of Theorem 5.3.1 builds on the bound on the difference between training dynamics of a two-layer fully-connected neural network trained with gradient descent and that of a linear model [71] early in training, with a modest generalization that this bound holds for isolated core and spurious features, as we justify in Appendix 9.4.1. At a high level, as the model is nearly linear in the initial $\nu_1 \cdot \frac{d \log d}{\eta}$ iterations, the contribution of the spurious feature $\boldsymbol{v}_s$ to the network output grows almost linearly with $(n_{c,s} - n_{c',s})\|\boldsymbol{v}_s\|^2$, at *every iteration* in the initial phase of training. Note that $n_{c,s} - n_{c',s}$ is the correlation between the spurious feature and the label c. When $n_{c,s} \gg n_{c',s}$, the spurious feature exists almost exclusively in the majority group of class c, and thus has a high correlation only with class c. In this case, if the magnitude of the spurious feature is significant, the contribution of the spurious feature to the model's output grows very rapidly, early in training. In particular, if $(n_{c,s} - n_{c',s})\|\boldsymbol{v}_s\|^2 \gg n_c\|\boldsymbol{v}_c\|^2$, the model's output is increasingly determined by the spurious feature, but not the core feature.

Remember from Sec. 5.2 that every example consists of a core and a spurious feature. As the effect of spurious features of the majority groups is amplified in the network output, the model's output will differ for examples in the majority and minority groups. The following corollary shows that the majority and minority groups are separable based on the network's output early in training. Notably, multiple minority groups with spurious features contained in majority groups of other classes are also separable.

Corollary 5.3.2 (**Separability of majority and minority groups**). *Suppose that for all classes, a majority group has at least K examples and a minority group has at most k examples. Then, under the assumptions of Theorem 5.3.1, examples in the majority and minority groups are separable based on the model's output, early in training. That is, for all $0 \le t \le \nu_1 \cdot \sqrt{\frac{d^{1-\alpha}}{\eta}}$, with high probability, the following holds for at least $1 - \mathcal{O}(d^{-\Omega(\alpha)})$ fraction of the training examples $\boldsymbol{x}_i$ in group $g_{c,s}$:*

If $g_{c,s}$ is in a majority group in class $c = 1$:

$$f(\boldsymbol{x}_i; \boldsymbol{W}_t, \boldsymbol{z}_t) \ge \frac{2\eta\zeta^2 t}{d}\left(\frac{\|\boldsymbol{v}_s\|^2(K-k)}{n} + \xi \pm O(d^{-\Omega(\alpha)})\right) + \rho(t, \phi, \Sigma), \tag{5.7}$$

If $g_{c,s}$ is in a minority group in class $c = 1$, but $g_{c',s}$ is a majority group in class $c' = -1$:

$$f(\boldsymbol{x}_i; \boldsymbol{W}_t, \boldsymbol{z}_t) \le \frac{2\eta\zeta^2 t}{d}\left(-\frac{\|\boldsymbol{v}_s\|^2(K-k)}{n} + \xi \pm O(d^{-\Omega(\alpha)})\right) + \rho(t, \phi, \Sigma), \tag{5.8}$$

where ρ is constant for all examples in the same class, $\xi \sim \mathcal{N}(0, \kappa)$ with

$$\kappa = \frac{1}{n}(\sum_c n_c^2 \sigma_c^2 \|\boldsymbol{v}_c\|^2)^{1/2} + \frac{1}{n}(\sum_s (n_{c,s} - n_{c',s})^2 \sigma_s^2 \|\boldsymbol{v}_s\|^2)^{1/2}$$

is the total effect of noise on the model.

Analogous statements holds for the class $c = -1$ by changing the sign and direction of the inequality.

The proof can be found in Appendix 9.4.3.2. Corollary 5.3.2 shows that when the majority group is considerably larger than the minority groups ($K \gg k$), the prediction of examples in the majority group move toward their label considerably faster, due to the contribution of the spurious feature. Hence, majority and minority groups can be separated from each other, early in training. Importantly, multiple minority groups can be also separated from each other, if their spurious feature exists in majority groups of other classes. Note that $K > k + |\xi|$ is the minimum requirement for the separation to happen. Separation is more significant when $K \gg k$ and when $\|\boldsymbol{v}_s\|$ is significant.

5.3.2 Network Exclusively Relies on Simple Spurious Features on Majority of Examples

Next, we analyze the second phase in early-time learning of a two-layer neural network. In particular, we show that if the noise-to-signal ratio of the spurious feature of the majority group of class c, i.e., $R_s = \sigma_s / \|\boldsymbol{v}_s\|$ is smaller than that of the core feature $R_c = \sigma_c / \|\boldsymbol{v}_c\|$, then the neural network's output is almost exclusively determined by the spurious feature and remain invariant to the core feature at $T = \nu_2 \cdot \frac{d \log d}{\eta}$, even though the core feature is more predictive of the class.

Theorem 5.3.3. *Under the assumptions of Theorem 5.3.1, if the classes are balanced, and the total size of the minority groups in class c is small, i.e., $\mathcal{O}(n^{1-\gamma})$ for some $\gamma > 0$, then there exists a constant $\nu_2 > 0$ such that at $T = \nu_2 \cdot \frac{d \log d}{\eta}$, for an example $\boldsymbol{x}_i$ in a majority group $g_{c,s}$, the contribution of the core feature to the model's output is at most:*

$$|f(\boldsymbol{v}_c; \boldsymbol{W}_T, \boldsymbol{z}_T)| \leq \sqrt{d} \frac{R_s}{\zeta R_c} + \mathcal{O}(n^{-\gamma} \sqrt{d}) + \mathcal{O}(d^{-\Omega(\alpha)}). \tag{5.9}$$

In particular if $\min\{R_c, 1\} \gg R_s$*, then the model's output is mostly indicated by the spurious* *feature instead of the core feature:*

$$|f(\boldsymbol{v}_s; \boldsymbol{W}_T, \boldsymbol{z}_T)| \geq \frac{\sqrt{d}}{2\zeta} \gg |f(\boldsymbol{v}_c; \boldsymbol{W}_T, \boldsymbol{z}_T)|. \tag{5.10}$$

The proof can be found in Appendix 9.4.3.3. The proof of Theorem 5.3.3 shows that at $T = \nu_2 \cdot \frac{d \log d}{\eta}$ where the linear model that closely mimics early-time learning dynamics of a two-layer FC neural network converges to its optimum parameters, the network has fully learned the spurious feature of the majority groups. At the same time, the contribution of the core feature to the network's output is at most proportional to R_s/R_c. Hence, if $R_s \ll R_c$, the core feature does not considerably contribute to the output of the neural network at T. That is, the network almost exclusively relies on the spurious feature of the majority group instead of the core feature which is more predictive of the class.

We note that our results in Theorem 5.3.1, Corollary 5.3.2, and Theorem 5.3.3 generalize to more than two classes and hold if the classes are imbalanced, as we will confirm by our experiments. Similar results can be shown for multi-layer fully connected and convolutional networks, following [71].

In Figure 5.1 illustrates, we empirically illustrate the above results during early-time training of LeNet-5 [96] on the Colored MNIST [8] dataset containing colored handwritten digits. Here, each digit is a class. The majority of digits in each class has a particular color, and the remaining digits are in four other colors. Figure 5.1b shows that the prediction of the network on the majority group is almost exclusively indicated by the color of the majority group, confirming Theorem 5.3.3. Figure 5.1c shows that the majority and minority groups are separable based on the network output, confirming Corollary 5.3.2.

Finally, note that by only learning the spurious feature, the neural network can shrinks the training loss on the majority of examples in class c to nearly zero and correctly classify them. Hence, the contribution of the spurious feature of the majority group of class c to the model's output is retained throughout the training. On the other hand, if minority groups are small, higher complexity functions that appear later in training overfit the minority groups, as observed by [156]. This results in a small training error but a poor worst-group generalization performance on the minorities.

Algorithm 4 SePArate early and REsample (Spare)

```
Require: Network $f(.,\boldsymbol{W})$, data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, loss function $\mathcal{L}$, iteration numbers $T_N, T_{init}$.
Ensure: Model $f$ trained without bias
  Stage 1: Early Bias Identification
  for $t = 0, \cdots, T_{init}$ do
    $\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t - \eta\nabla\mathcal{L}(\boldsymbol{W}_t; \mathcal{D})$
  end for
  for every class $c \in \mathcal{C}$ with examples $V_c$ do
    Identify $\lambda$, # of clusters $k$ via Silhouette analysis
    Cluster $V_c$ into $\{V_{c,j}\}_{j=1}^k$ based on $f(\boldsymbol{x}_i; \boldsymbol{W}_t)$
    Weight every $\boldsymbol{x}_i \in V_{c,j}$ by $w_i = 1/|V_{c,j}|$, $p_i = w_i^\lambda / \sum_i w_i^\lambda$
  end for
  Stage 2: Learning without Bias
  for $t = 0, \cdots, T_N$ do
    Sample a mini-batch $\mathcal{M}_t = \{(\boldsymbol{x}_i, y_i)\}_i$ with probabilities $p_i$
    $\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta\nabla\mathcal{L}(\boldsymbol{W}_t; \mathcal{M}_t)$.
  end for
```

5.4 Eliminating Spurious Bias Early in Training

Next, we rely on our theoretical insights from Sec. 5.3 to mitigate spurious correlations while training neural networks. To do so, we first leverage the model's output to separate majority and minority groups. Then, we apply importance sampling to amplify the effect of core features over the spurious features, by making the groups relatively balanced.

Stage 1: Separating the Groups Early in Training. Corollary 5.3.2 shows that majority and minority groups are separable based on the network's output. To identify the majority and minority groups, we cluster examples V_c in every class $c \in \mathcal{C}$ based on the output of the network, during the first few epochs. We determine the number of clusters via silhouette analysis [151]. In doing so, we can separate majority and minority groups in each class. Any clustering algorithm such as k-means or k-median clustering can be applied to separate the groups. While k-means easily scales to medium-sized datasets, k-median is more suitable for very large datasets, as it can be formulated as a submodular maximization problem [203] for which fast and scalable distributed [124, 125] and streaming [14] algorithms are available. We note that following Corollary 5.3.2 we cluster the entire network output and not only the class confidence, which yields superior results.

Stage 2: Making the Groups More Balanced via Importance Sampling. To alleviate the spurious correlations and enable effective learning of the core features, we employ an importance sampling method on examples in each class to upsample examples in the smaller clusters and downsample examples in the larger clusters. To do so, we assign every example $i \in V_{c,j}$ a weight given by the size of the cluster it belongs to, i.e., $w_i = 1/|V_{c,j}|$. Then we sample examples in every mini-batch with probabilities equal to $p_i = w_i^\lambda / \sum_i w_i^\lambda$, where λ can be determined based on the average silhouette score of clusters in each class. A higher average silhouette score indicates that clusters are more separated. In this case, groups can be accurately identified and we can balance the groups using $\lambda = 1$. However, when clusters are not well separated (lower silhouette score), some examples from the majority group are spread in smaller clusters. In this case, sampling less from the large clusters is enough to balance the groups, as the majority groups are sampled when we upsample the small clusters. Here, we can balance the groups using $\lambda = 2$. Empirically, we found that $\lambda = 1$ or 2 is enough to effectively mitigate the spurious correlation in all our experiments. Note that our importance sampling method does not increase the size of the training data, and only changes the data distribution. Hence, it does not increase the training time. The pseudocode is illustrated in Alg. 4.

5.5 Experiments

In this section, we evaluate the effectiveness of Spare in finding spurious correlations early in training, and eliminating while training on various spurious benchmark datasets. Then, we apply Spare to discover and mitigate spurious correlations in Restricted ImageNet to confirm its broader applicability.

Table 5.1: Worst-group and average accuracy (%) of training with SPARE vs. state-of-the-art algorithms, on datasets with spurious correlations. CB, GB indicate balancing classes and groups, respectively. Numbers indicated with * are from [229]. SPARE achieves a superior performance much faster. † We couldn't replicate DFR's result on CMNIST due to the lack of experimental details in the original paper, so we cite the only reported number for reference and leave other entries blank.

	Group	Train	**CMNIST**		**Waterbirds**		**CelebA**	
	Info	Cost	Worst-group	Average	Worst-group	Average	Worst-group	Average
ERM	×	1x	$0.0^{*}_{\pm 0.0}$	$20.1^{*}_{\pm 0.2}$	$62.6^{*}_{\pm 0.3}$	$97.3^{*}_{\pm 1.0}$	$47.7^{*}_{\pm 2.1}$	$94.9^{*}_{\pm 0.3}$
CB	×	1x	$0.0_{\pm 0.0}$	$23.7_{\pm 3.1}$	$62.8_{\pm 1.6}$	$97.1_{\pm 0.1}$	$46.1_{\pm 1.5}$	$95.2_{\pm 0.4}$
EIIL	×	1x	$72.8^{*}_{\pm 6.8}$	$90.7^{*}_{\pm 0.9}$	$77.2^{*}_{\pm 1.0}$	$96.5^{*}_{\pm 0.2}$	$81.7^{*}_{\pm 0.8}$	$85.7^{*}_{\pm 0.1}$
PGI	×	1x	$73.5^{*}_{\pm 1.8}$	$88.5^{*}_{\pm 1.4}$	$79.5^{*}_{\pm 1.9}$	$95.5^{*}_{\pm 0.8}$	$85.3^{*}_{\pm 0.3}$	$87.3^{*}_{\pm 0.1}$
GEORGE	×	2x	$76.4^{*}_{\pm 2.3}$	$89.5^{*}_{\pm 0.3}$	$76.2^{*}_{\pm 2.0}$	$95.7^{*}_{\pm 0.5}$	$54.9^{*}_{\pm 1.9}$	$94.6^{*}_{\pm 0.2}$
LfF	×	2x	$0.0^{*}_{\pm 0.0}$	$25.0^{*}_{\pm 0.5}$	$78.0^{*}_{N/A}$	$91.2^{*}_{N/A}$	$77.2^{*}_{N/A}$	$85.1^{*}_{N/A}$
CIM	×	2x	$0.0^{*}_{\pm 0.0}$	$36.8^{*}_{\pm 1.3}$	$77.2^{*}_{N/A}$	$95.6^{*}_{N/A}$	$83.6^{*}_{N/A}$	$90.6^{*}_{N/A}$
JTT	×	5x-6x	$74.5^{*}_{\pm 2.4}$	$90.2^{*}_{\pm 0.8}$	$83.8^{*}_{\pm 1.2}$	$89.3^{*}_{\pm 0.7}$	$81.5^{*}_{\pm 1.7}$	$88.1^{*}_{\pm 0.3}$
CnC	×	2x-12x	$77.4^{*}_{\pm 3.0}$	$90.9^{*}_{\pm 0.6}$	$88.5^{*}_{\pm 0.3}$	$90.9^{*}_{\pm 0.1}$	$88.8^{*}_{\pm 0.9}$	$89.9^{*}_{\pm 0.5}$
SPARE	×	1x	$\mathbf{83.0}_{\pm 1.7}$	$\mathbf{91.8}_{\pm 0.7}$	$\mathbf{89.8}_{\pm 0.6}$	$94.2_{\pm 1.6}$	$\mathbf{90.3}_{\pm 0.3}$	$91.1_{\pm 0.1}$
SSA	validation	1.5x-5x	$0.0_{\pm 0.0}$	$47.9_{\pm 14.4}$	$89.0_{\pm 0.6}$	$92.2_{\pm 0.9}$	$\mathbf{89.8}_{\pm 1.3}$	$92.8_{\pm 0.1}$
DFR^{Tr}_{Tr}	training sub.	1x	-	-	$90.4_{\pm 1.5}$	$94.1_{\pm 0.5}$	$80.1_{\pm 1.1}$	$89.7_{\pm 0.4}$
DFR^{Val}_{Tr}	validation	1x	$80.4^{\dagger}_{\pm 1.1}$	-	$\mathbf{91.8}_{\pm 2.6}$	$93.5_{\pm 1.4}$	$87.3_{\pm 1.0}$	$90.2_{\pm 0.8}$
GB	training	1x	$\mathbf{82.2}_{\pm 1.0}$	$91.7_{\pm 0.6}$	$86.3_{\pm 0.3}$	$93.0_{\pm 1.5}$	$85.0_{\pm 1.1}$	$92.7_{\pm 0.1}$
GDRO	training	1x	$78.5_{\pm 4.5}$	$90.6_{\pm 0.1}$	$89.9^{*}_{\pm 0.6}$	$92.0^{*}_{\pm 0.6}$	$88.9^{*}_{\pm 1.3}$	$93.9^{*}_{\pm 0.1}$

5.5.1 Mitigating Spurious Correlations in Benchmark Datasets

We first evaluate the effectiveness of SPARE in alleviating spurious correlations on spurious benchmarks. The reported results are averaged over three runs with different model initializations.

Datasets & Models. (1) CMNIST [8] contains colored handwritten digits derived from MNIST [96]. We follow the challenging 5-class setting in [229] where every two digits form one class and 99.5% of training examples in each class are spuriously correlated with a distinct color. We use a 5-layer CNN (LeNet-5 [96]) for CMNIST. (2) Waterbirds [155] contains two classes (landbird vs. waterbird) and the background (land or water) is the spurious feature. Majority groups are (waterbird, water) and (landbird, land). (3) CelebA [110] is another most commonly used benchmark for spurious correlations. Following [155], we consider the hair color (blond vs. non-blond) as the class labels and gender (male or female) as the spurious feature. The majority groups are (blond, female) and (non-blond male). For both Waterbirds and CelebA, we follow the standard settings used in the previous work to train a ResNet-50 model [67] pretrained on ImageNet provided by the Pytorch library [141]. More details about the datasets and the experimental settings can be found in Section 9.4.4.

Baselines. We compare Spare with the state-of-the-art methods for eliminating spurious correlations in Table 5.1, in terms of both worst-group accuracy, i.e., the minimum accuracy across all groups, and average accuracy. We use adjusted average accuracy for Waterbirds, i.e., the average accuracy over groups weighted by their size. This is consistent with prior work, and is done because the validation and test sets are group-balanced while the training set is skewed. GB (Group Balancing) and GDRO [155] use the group label of all training examples, and SSA [133] uses the group labels of the validation data. DFR [90] uses a group-balanced data drawn from either validation (DFR$^{Tr}_{Tr}$) or training (DFR$^{Val}_{Tr}$) data. The rest of the methods infer the group labels without using such information.

Spare outperforms SOTA algorithms, including those that require group information. Table 5.1 shows that compared to baselines that do not use the group labels, Spare obtains the *highest* worst-group accuracy, while maintaining high average accuracy. In particular, Spare consistently outperforms the best baselines, CnC [229] and JTT [106], on worst-group and average accuracy while having up to 12x lower computational cost. Notably, Spare performs comparably to those that use the group information, and even achieves a better worst-group accuracy on CMNIST and CelebA and has a comparable worst-group but higher average accuracy on the Waterbirds. Note that DFR$^{Val}_{Tr}$ trains on group-balanced validation data, while Spare does not. As group labels are unavailable in real-world datasets, methods that do not rely on group labels are more practical. Among such methods, Spare has a superior performance and easily scales to large datasets. Notably, Spare finds the groups, at epoch 2 for CMNIST and Waterbirds, and at epoch 1 for CelebA.

Spare reaches SOTA performance under extreme group imbalance. Many state-of-the-art algorithms that can successfully eliminate spurious correlations in the Waterbirds and CelebA, severely fail on CMNIST, by providing as low as 0% worst-group accuracy. In

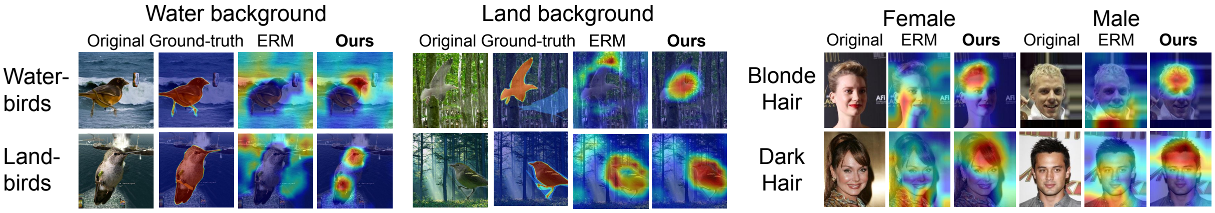


Figure 5.2: GradCAM Visualization. Warmer colors correspond to the pixels that are weighed more in making the final classification. SPARE allows learning the core features instead of spurious ones.

CMNIST, every class has a very large majority and *four* very small minority groups, and there is a very strong spurious correlation between the color of the majority group and the corresponding class. Here, the small size of the minority groups makes it difficult to infer the groups based on loss (LfF [132]), data augmentation (CIM [178]), or semi-supervised learning (SSA [133]). Besides, state-of-the-art methods that partition every class into only two groups, namely EIIL [43], PGI [6], CnC [229], and JTT [106], fail to balance the minority groups. This is because the minority groups need to be extensively upweighted or upsampled to make a balance with the majority group due to their small sizes, and extensive upweighting or upsampling them as a whole exaggerates the small differences between the original size of the minority groups and makes them imbalanced w.r.t. each other. This yields an inferior worst-group accuracy. In contrast, SPARE finds multiple minority clusters via silhouette analysis (see Figure 5.1c). By importance sampling from each cluster based on its size, SPARE can successfully balance the groups and achieve state-of-the-art worst-group and average accuracy.

GradCAM. Fig. 5.2 compares GradCAM [162] visualizations depicting saliency maps for samples from Waterbirds with water and land backgrounds (left), and from CelebA with different genders (right), when ResNet50 is trained by ERM vs. SPARE. Warmer colors indicate the pixels that the model considered more important for making the final classification, based on gradient activations. We see that training with SPARE allows the model to learn the core feature, instead of the spurious features.

Table 5.2: Identifying groups in Waterbirds, based on the output of different networks early in training.

Model	Epoch	WG Acc	Avg Acc
RN18	4	$89.5_{\pm 0.8}$	$96.6_{\pm 0.3}$
RN50	4	$89.1_{\pm 2.1}$	$95.9_{\pm 0.7}$
Wide RN50	5	$90.8_{\pm 0.5}$	$95.9_{\pm 0.7}$
DenseNet121	5	$89.3_{\pm 1.5}$	$96.3_{\pm 0.6}$
Pretrained RN50	1	$89.8_{\pm 0.6}$	$94.2_{\pm 1.6}$

Table 5.3: Average Silhouette scores of clusters in different classes, and the corresponding importance sampling power (λ) used for each class.

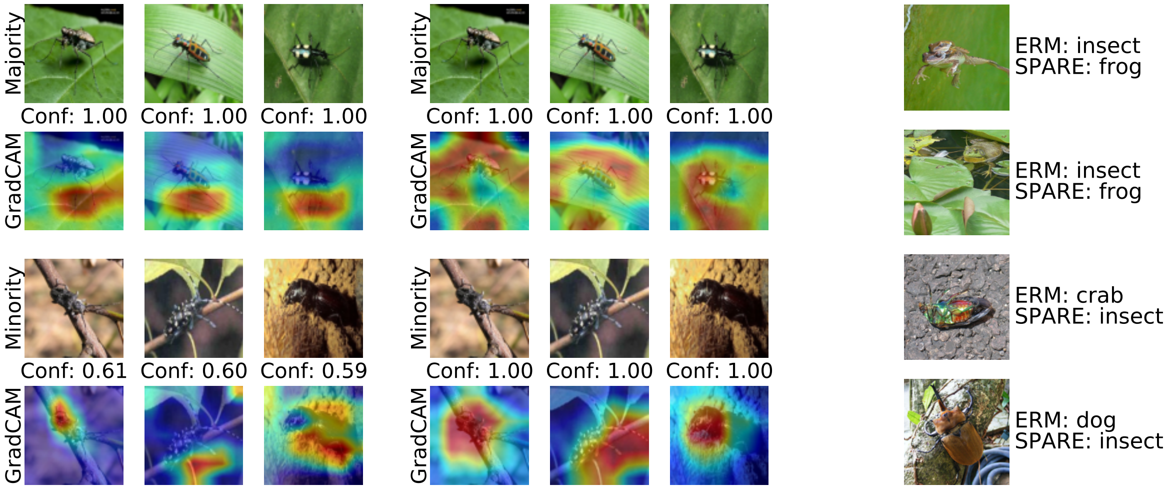
Dataset	Silhouette score	Sampling power (λ)
CMNIST	between 0.991-0.997	[1, 1, 1, 1, 1]
Waterbirds	[0.886, 0.758]	[2, 2]
CelebA	[0.924, 0.757]	[1, 2]

5.5.2 Ablation Studies

Next, we show that Spare effectively separates majority and minority groups regardless of the network architecture early in training and explain how we determine cluster importance using silhouette scores without group information.

Network Structure. First, we show Spare can separate majority and minority groups independent of the network architecture, early in training. To do so, we cluster output of ResNets of varying depths and widths, namely, ResNet-18, ResNet-50, Wide ResNet-50 [225], as well as DenseNet121 [73], and apply importance sampling to the clusters to train a pretrained ResNet-50 model on Waterbirds (same setting as in Table 5.1). Table 5.2 shows the worst-group and adjusted average accuracy. We see that all the networks can successfully separate the groups within the first 5 epochs. This confirms the effectiveness of simplicity bias in identifying groups in the early training phase. Details of the experiments can be found in Section 9.4.4.

Importance Sampling Power (λ). Next, we explain how we determine the importance of different clusters using silhouette scores. Table 5.3 presents the average silhouette score for each class in different datasets. A higher average silhouette score indicates that clusters are well separated, such as in CMNIST and the female class in CelebA. This means we can accurately identify both the majority and minority groups. However, when clusters are not



(a) Insects in ImageNet, Epoch 8.

(b) Insects in ImageNet, End.

(c) SPARE corrects spurious correlation.

Figure 5.3: Spurious correlation between "green leaf" & "insect" in Restricted ImageNet found by SPARE.

clearly separated (lower silhouette scores), some examples from the majority group get mixed up with the smaller clusters. As a result, we sample even fewer examples from the larger clusters. When clusters are well separated, we use $\lambda = 1$ to ensure equal treatment of groups. However, for less separable clusters, using $\lambda = 2$ helps achieve group balance.

5.5.3 Discovering and Mitigating Spurious Correlations in Restricted ImageNet

Finally, we show the applicability of SPARE to discover and mitigate spurious correlations in more realistic settings. We use Restricted ImageNet [178], a 9-superclass subset of ImageNet, to train ResNet-50 from scratch. We applied SPARE to cluster the model's output in the first 10 epochs, and inspected the clusters as described below. However, we found that the results are not very sensitive to the choice of the initial epoch. See Appendix 9.4.5 for more details on the dataset and experiment.

Frog vs. Insect. By inspecting the clusters with the highest fraction of misclassified examples to another class, we find that many Frog images are misclassified as Insects. Figure 5.3a shows examples from the two groups SPARE finds for the Insect class at epoch 8. GradCAM reveals an obvious spurious correlation between “green leaf" and the insect class that is maintained until the end of the training, as illustrated in Figure 5.3b. We also observe

Table 5.4: Mitigating spurious in Restricted ImageNet.

	Test Acc	Insect Minority	Frog Minority
ERM	96.0%	83.3%	96.2%
CB	95.9%	87.5% ↑	96.2%−
EIIL	93.1%	76.0% ↓	90.4% ↓
Spare	95.4%	86.3% ↑	98.1% ↑

a large gap between the confidence of examples in the two groups. This indicates that the model has learned the spurious feature early in training.

Next, we apply importance sampling with $\lambda = 2$ (silhouette score < 0.9), to alleviate the spurious correlation. Here, as the underlying group labels are not available, GroupDRO and GB are not applicable. Besides, state-of-the-art methods such as JTT and CnC are prohibitively slow to apply to ImageNet (see training cost in Table 5.1). Hence, we only compare Spare with Class Balancing (CB) and EIIL. Table 5.4 shows that Spare can successfully improve the accuracy of both insect and frog minorities by 3% and 2% respectively, with the slightest drop in the total accuracy. Note that examples in Frog minority group were classified as Insect, due to “green leaf" in their background. Figure 5.3c shows examples of Frog and Insect minorities that are correctly classified after training with Spare. In contrast, CB improves the accuracy only on Insect minority. Note that while CB upsamples both classes, it cannot improve the accuracy on Frog minority examples that are misclassified as Insect due to the spurious correlation. We also see that EIIL drops the accuracy on minorities as well as the total accuracy. This is mainly because EIIL separates and upweights many examples that are misclassified by ERM, as we report in Section 9.4.5. Extensive upweighting of such examples harms the performance. We expect this effect to be even more severe for methods like JTT, which directly find the misclassified examples as the minority group. In contrast, Spare better separates the groups and mitigates the spurious correlation by balancing them.

5.6 Conclusion

In this chapter, I studied how simple spurious features are learned during training neural networks with gradient methods. In particular, I analyzed a two-layer fully-connected neural network and showed that large groups of examples with spurious features are separable based on the model's output, early in training. If spurious features have a small enough noise-to-signal ratio, the network's output on a large number of examples will be almost exclusively determined by the spurious features and will be nearly invariant to the core features. Based on the above theoretical insights, I proposed Spare, which separates majority and minority groups by clustering the model output early in training. Then, it applies importance sampling based on the cluster sizes to make the groups relatively balanced. I showed that Spare achieves state-of-the-art worst-group accuracy on various datasets, and is highly scalable. I also demonstrated the applicability of Spare in more realistic settings, to discover and mitigate spurious correlations from Restricted ImageNet.

CHAPTER 6

PDE: Data-efficient and Robust Training against Spurious Correlations

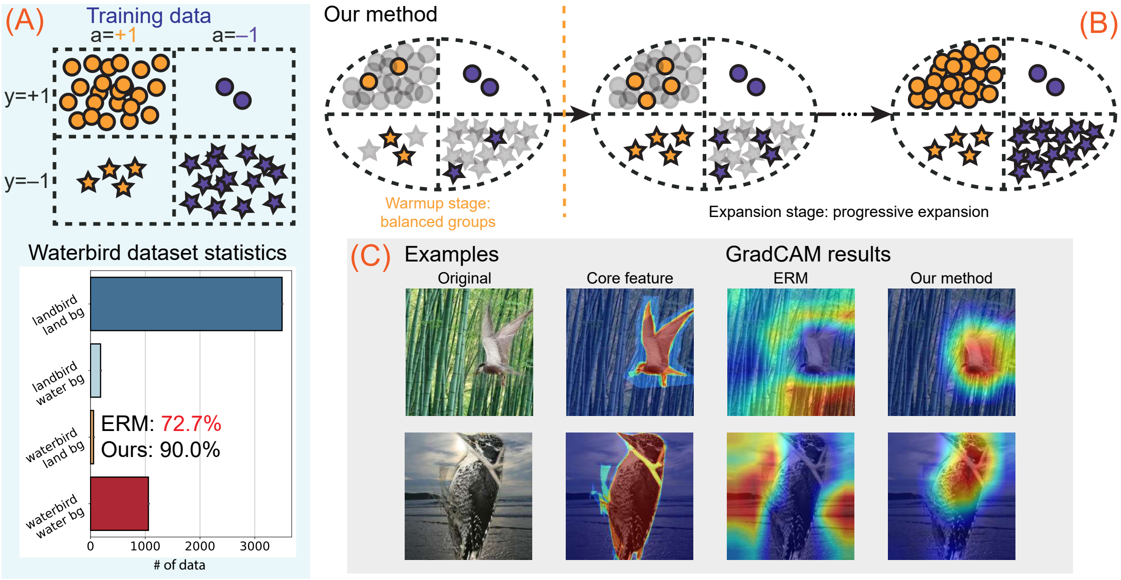


Figure 6.1: An overview of the problem, our proposed solution, and the resultant outcomes. (A) We demonstrate the data distribution and provide an example of the statistics of Waterbirds. (B) The overall procedure of PDE. (C) We use GradCAM [162] to show the attention of the model trained with PDE as compared to ERM. While ERM focuses on the background, PDE successfully trains the model to capture the birds.

Despite the remarkable performance of deep learning models, recent studies [155, 156, 76, 65, 211, 215, 218, 85] have identified their vulnerability to spurious correlations in data distributions. A spurious correlation refers to an easily learned feature that, while unrelated to the task at hand, appears with high frequency within a specific class. For instance, waterbirds frequently appear with water backgrounds, and landbirds with land backgrounds. When training with empirical risk minimization (ERM), deep learning models tend to exploit such correlations and fail to learn the more subtle features genuinely correlated with the true labels, resulting in poor generalization performance on minority data (e.g., waterbirds with land backgrounds as shown in Figure 6.1). This observation raises a crucial question: *Does the model genuinely learn to classify birds, or does it merely learn to distinguish land from water?* The issue is particularly concerning because deep learning models are being deployed in critical applications such as healthcare, finance, and autonomous vehicles, where we require a reliable predictor.

Researchers formalized the problem by considering examples with various combinations of core features (e.g., landbird/waterbird) and spurious features (e.g., land/water backgrounds) as different *groups*. The model is more likely to make mistakes on certain groups if it learns the spurious feature. The objective therefore becomes balancing and improving performance across all groups. Under this formulation, we can divide the task into two sub-problems: (1) accurately identifying the groups, which are not always known in a dataset, and (2) effectively using the group information to finally improve the model's robustness. While numerous recent works [132, 106, 43, 7, 178, 229] focus on the first sub-problem, the second sub-problem remains understudied. The pioneering work [155] still serves as the best guidance for utilizing accurate group information. In this paper, we focus on the second sub-problem and aim to provide a more effective and efficient algorithm to utilize the group information. It is worth noting that the theoretical understanding of spurious correlations lags behind the empirical advancements in mitigating spurious features. Existing theoretical studies [156, 35, 211, 221] are limited to the setting of simple linear models and data distribution that are less reflective of real application scenarios.

We begin by theoretically examining the learning process of spurious features when training a two-layer nonlinear convolutional neural network (CNN) on a corresponding data model that captures the influence of spurious correlations. We illustrate that the learning of spurious features swiftly overshadows the learning of core features from the onset of training when groups are imbalanced and spurious features are more easily learned than core features. Based upon our theoretical understanding, we propose Progressive Data Expansion (**PDE**), a neat and novel training algorithm that efficiently uses group information to enhance the model's robustness against spurious correlations. Existing approaches, such as GroupDRO [155] and upsampling techniques [106], aim to balance the data groups in each batch throughout the training process. In contrast, we employ a small balanced warm-up subset only at the beginning of the training. Following a brief period of balanced training, we progressively expand the warm-up subset by adding small random subsets of the remaining training data until using all of them, as shown in the top right of Figure 6.1. Here, we utilize the momentum from the warm-up subset to prevent the model from learning spurious features when adding new data. Empirical evaluations on both synthetic and real-world benchmark data validate our theoretical findings and confirm the effectiveness of PDE. Additional ablation studies also demonstrate the significance and impact of each component within our training scheme. In summary, our contributions are highlighted as follows:

- We provide a theoretical understanding of the impact of spurious correlations beyond the linear setting by considering a two-layer nonlinear CNN.
- We introduce PDE, a theory-inspired approach that effectively addresses the challenge posed by spurious correlations.
  - PDE achieves the best performance on benchmark vision and language datasets for models including ResNets and Transformers. On average, it outperforms the state-of-the-art method by 2.8% in terms of worst-group accuracy.
  - PDE enjoys superior training efficiency, being 10× faster than the state-of-the-art methods.

6.1 Why is Spurious Correlation Harmful to ERM?

In this section, we simplify the intricate real-world problem of spurious correlations into a theoretical framework. We provide analysis on two-layer nonlinear CNNs, extending beyond the linear setting prevalent in existing literature on this subject. Under this framework, we formally present our theory concerning the training process of empirical risk minimization (ERM) in the presence of spurious features. These theoretical insights motivate the design of our algorithm.

6.1.1 Empirical Risk Minimization

We begin with the formal definition of the ERM-based training objective for a binary classification problem. Consider a training dataset $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the input and $y \in \{\pm 1\}$ is the output label. We train a model $f(\mathbf{x}; \mathbf{W})$ with weight $\mathbf{W}$ to minimize the empirical loss function:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{N} \textstyle\sum_{i=1}^N \ell\big(y_i f(\mathbf{x}_i; \mathbf{W})\big), \tag{6.1}$$

where ℓ is the logistic loss defined as $\ell(z) = \log(1 + \exp(-z))$. The empirical risk minimizer refers to $\mathbf{W}^*$ that minimizes the empirical loss: $\mathbf{W}^* := \arg\min_{\mathbf{W}} \mathcal{L}(\mathbf{W})$. Typically, gradient-based optimization algorithms are employed for ERM. For example, at each iteration t, gradient descent (GD) has the following update rule:

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla \mathcal{L}(\mathbf{W}^{(t)}). \tag{6.2}$$

Here, $\eta > 0$ is the learning rate. In the next subsection, we will show that even for a relatively simple data model which consists of core features and spurious features, vanilla ERM will fail to learn the core features that are correlated to the true label.

6.1.2 Data Distribution with Spurious Correlation Fails ERM

Previous work such as [156] considers a data model where the input consists of core feature, spurious feature and noise patches at fixed positions, i.e., $\mathbf{x} = [\mathbf{x}_{\text{core}}, \mathbf{x}_{\text{spu}}, \mathbf{x}_{\text{noise}}]$. In real-world applications, however, features in an image do not always appear at the same pixels. Hence, we consider a more realistic data model where the patches do not appear at fixed positions.

Definition 6.1.1 (Data model)**.** A data point $(\mathbf{x}, y, a) \in (\mathbb{R}^d)^P \times \{\pm 1\} \times \{\pm 1\}$ is generated from the distribution $\mathcal{D}$ as follows.

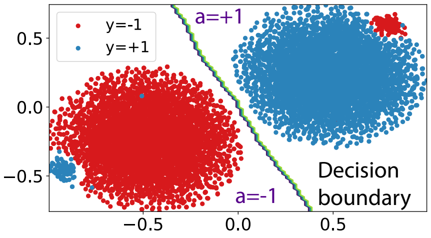


Figure 6.2: Visualization of the data.

- Randomly generate the true label $y \in \{\pm 1\}$.
- Generate spurious label $a \in \{\pm y\}$, where $a = y$ with probability $\alpha > 0.5$.
- Generate $\mathbf{x}$ as a collection of P patches: $\mathbf{x} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(P)}) \in (\mathbb{R}^d)^P$, where
  - **Core feature.** One and only one patch is given by $\beta_c \cdot y \cdot \boldsymbol{v}_c$ with $\|\boldsymbol{v}_c\|_2 = 1$.
  - **Spurious feature.** One and only one patch is given by $\beta_s \cdot a \cdot \boldsymbol{v}_s$ with $\|\boldsymbol{v}_s\|_2 = 1$ and $\langle \boldsymbol{v}_c, \boldsymbol{v}_s \rangle = 0$.
  - **Random noise.** The rest of the $P - 2$ patches are Gaussian noises $\boldsymbol{\xi}$ that are independently drawn from $N(0, (\sigma_p^2/d) \cdot \mathbf{I}_d)$ with σ_p as an absolute constant.

  And $0 < \beta_c \ll \beta_s \in \mathbb{R}$.

Similar data models have also been considered in recent works on feature learning [10, 239, 37, 81], where the input data is partitioned into feature and noise patches. We extend their data models by further positing that certain feature patches might be associated with the spurious label instead of the true label. In the rest of the paper, we assume $P = 3$ for simplicity. With the given data model, we consider the training dataset $S = \{(\mathbf{x}_i, y_i, a_i)\}_{i=1}^N$ and let S be partitioned into large group S_1 and small group S_2 such that S_1 contains all the training data that can be correctly classified by the spurious feature, i.e., $a_i = y_i$, and S_2 contains all the training data that can only be correctly classified by the core feature, i.e., $a_i = -y_i$. We denote $\hat{\alpha} = \frac{|S_1|}{N}$ and therefore $1 - \hat{\alpha} = \frac{|S_2|}{N}$.

Visualization of our data. In Figure 6.2, we present the visualization in 2D space of the higher-dimensional data generated from our data model using t-SNE [190], where data within each class naturally segregate into large and small groups. The spurious feature is sufficient for accurate classification of the larger group data, but will lead to misclassification of the small group data.

6.1.3 Beyond Linear Models

We consider a two-layer nonlinear CNN defined as follows:

$$f(\mathbf{x};\mathbf{W}) = \textstyle\sum_{j\in[J]}\sum_{p=1}^{P}\sigma\big(\langle \mathbf{w}_j,\mathbf{x}^{(p)}\rangle\big), \tag{6.3}$$

where $\mathbf{w}_j \in \mathbb{R}^d$ is the weight vector of the j-th filter, J is the number of filters (neurons) of the network, and $\sigma(z) = z^3$ is the activation function. $\mathbf{W} = [\mathbf{w}_1,\dots,\mathbf{w}_J] \in \mathbb{R}^{d\times J}$ denotes the weight matrix of the CNN. Similar two-layer CNN architectures are analyzed in in [37, 81] but for different problems, where the cubic activation serves a simple function that provides non-linearity. Similar to **(author?)** [81, 30], we assume a mild overparameterization of the CNN with $J = \text{polylog}(d)$. We initialize $\mathbf{W}^{(0)} \sim \mathcal{N}(0,\sigma_0^2)$, where $\sigma_0^2 = \text{polylog}(d)/d$. Due to the CNN structure, our analysis can handle data models where each data can have an arbitrary order of patches while linear models fail to do so.

6.1.4 Understanding the Training Process with Spurious Correlation

In this subsection, we formally introduce our theoretical result on the training process of the two-layer CNN using gradient descent in the presence of spurious features. We first define the performance metrics. A frequently considered metric is the test accuracy: $\text{Acc}(\mathbf{W}) = \mathbb{P}_{(\mathbf{x},y,a)\sim\mathcal{D}}\big[\text{sgn}(f(\mathbf{x};\mathbf{W})) = y\big]$. With spurious correlations, researchers are more interested in the worst-group accuracy:

$$\text{Acc}_{\text{wg}}(\mathbf{W}) = \min_{y\in\{\pm1\},a\in\{\pm1\}} \mathbb{P}_{(\mathbf{x},y,a)\sim\mathcal{D}}\big[\text{sgn}(f(\mathbf{x};\mathbf{W})) = y\big],$$

which accesses the worst accuracy of a model among all groups defined by combinations of y and a. We then summarize the learning process of ERM in the following theorem. Our analysis focuses on the learning of spurious and core features, represented by the growth of $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v}_s \rangle$ and $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v}_c \rangle$ respectively:

Theorem 6.1.2. *Consider the training dataset $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the distribution in Definition 6.1.1. Consider the two-layer nonlinear CNN model as in (6.3) initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$. After training with GD in (6.2) for $T_0 = \tilde{\Theta}\big(1/(\eta\beta_s^3\sigma_0)\big)$ iterations, for all $j \in [J]$ and $t \in [0, T_0)$, we have*

$$\tilde{\Theta}(\eta)\beta_s^3(2\hat{\alpha}-1) \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2 \le \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s \rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle \le \tilde{\Theta}(\eta)\beta_s^3\hat{\alpha} \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2, \tag{6.4}$$

$$\tilde{\Theta}(\eta)\beta_c^3\hat{\alpha} \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle^2 \le \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c \rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle \le \tilde{\Theta}(\eta)\beta_c^3 \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle^2. \tag{6.5}$$

After training for T_0 iterations, with high probability, the learned weight has the following properties: (1) it learns the spurious feature $\boldsymbol{v}_s$: $\max_{j\in[J]}\langle \mathbf{w}_j^{(T)}, \boldsymbol{v}_s \rangle \ge \tilde{\Omega}(1/\beta_s)$; (2) it almost does not learn the core feature $\boldsymbol{v}_c$: $\max_{j\in[J]}\langle \mathbf{w}_j^{(T)}, \boldsymbol{v}_c \rangle = \tilde{\mathcal{O}}(\sigma_0)$.

Discussion. The detailed proof is deferred to Appendix 9.5.5, and we provide intuitive explanations of the theorem as follows. A larger value of $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v} \rangle$ for $\boldsymbol{v} \in \{\boldsymbol{v}_s, \boldsymbol{v}_c\}$ implies better learning of the feature vector $\boldsymbol{v}$ by neuron $\mathbf{w}_i$ at iteration t. As illustrated in (6.4) and (6.5), the updates for both spurious and core features are non-zero, as they depend on the squared terms of themselves with non-zero coefficients, while the growth rate of $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v}_s \rangle$ is significantly faster than that of $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v}_c \rangle$. Consequently, the neural network rapidly learns the spurious feature but barely learns the core feature, as it remains almost unchanged from initialization as compared to the spurious feature.

We derive the neural network's prediction after training for T_0 iterations. For a randomly generated data example $(\mathbf{x}, y, a) \sim \mathcal{D}$, the neural network's prediction is given by $\operatorname{sgn}\big(f(\mathbf{x}; \mathbf{W})\big) = \operatorname{sgn}\big(\sum_{j\in[J]} \big(y\beta_c^3\langle \mathbf{w}_j, \boldsymbol{v}_c \rangle^3 + a\beta_s^3\langle \mathbf{w}_j, \boldsymbol{v}_s \rangle^3 + \langle \mathbf{w}_j, \xi \rangle^3\big)\big)$. Since the term $\beta_s^3 \max_{j\in[J]}\langle \mathbf{w}_j, \boldsymbol{v}_s \rangle^3$ dominates the summation, the prediction will be $\operatorname{sgn}(f(\mathbf{x}; \mathbf{W})) = a$. Consequently, we obtain the test accuracy as $\mathrm{Acc}(\mathbf{W}) = \alpha$, since $a = y$ with probability α, and the model accurately classifies the large group. However, when considering the small group and examining examples where $y \neq a$, the models consistently make errors, resulting in

$\mathrm{Acc}_{wg}(\mathbf{W}) = 0$. To circumvent this poor performance on worst-group accuracy, an algorithm that can avoid learning the spurious feature is in demand.

6.2 Theory-Inspired Two-Stage Training Algorithm

In this section, we introduce Progressive Data Expansion (PDE), a novel two-stage training algorithm inspired by our analysis to enhance robustness against spurious correlations. We begin with illustrating the implications of our theory, where we provide insights into the data distributions that lead to the rapid learning of spurious features and clarify scenarios under which the model remains unaffected.

6.2.1 Theoretical Implications

Notably in Theorem 6.1.2, the growth of the two sequences $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v}_s \rangle$ in (6.4) and $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v}_c \rangle$ in (6.5) follows the formula $x_{t+1} = x_t + \eta A x_t^2$, where x_t represents the inner product sequence with regard to iteration t and A is the coefficient containing $\hat{\alpha}$, β_c or β_s. This formula is closely related to the analysis of tensor power methods [10]. In simple terms, when two sequences have slightly different growth rates, one of them will experience much faster growth in later times. As we will show below, the key factors that determine the drastic difference between spurious and core features in later times are the group size $\hat{\alpha}$ and feature strengths β_c, β_s.

- **When the model learns spurious feature ($\beta_c^3 < \beta_s^3(2\hat{\alpha} - 1)$).** We examine the lower bound for the growth of $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v}_s \rangle$ in (6.4) and the upper bound for the growth of $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v}_c \rangle$ in (6.5) in Theorem 6.1.2. If $\beta_c^3 < \beta_s^3(2\hat{\alpha} - 1)$, we can employ the tensor power method and deduce that the spurious feature will be learned first and rapidly. The condition on data distribution imposes two necessary conditions: $\hat{\alpha} > 1/2$ (groups are imbalanced) and $\beta_c < \beta_s$ (the spurious feature is stronger). This observation is consistent with real-world datasets, such as the Waterbirds dataset, where $\hat{\alpha} = 0.95$ and the background is much easier to learn than the intricate features of the birds.

- **When the model learns core feature ($\beta_c > \beta_s$).** However, if we deviate from the aforementioned conditions and consider $\beta_c > \beta_s$, we can examine the lower bound for the growth of $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v}_c \rangle$ in (6.5) and the upper bound for the growth of $\langle \mathbf{w}_i^{(t)}, \boldsymbol{v}_s \rangle$ in (6.4). Once again, we apply the tensor power method and determine that the model will learn the core feature rapidly. In real-world datasets, this scenario corresponds to cases where the core feature is not only significant but also easier to learn than the spurious feature. Even for imbalanced groups with $\hat{\alpha} > 1/2$, the model accurately learns the core feature. Consequently, enhancing the coefficients of the growth of the core feature allows the model to tolerate imbalanced groups. We present verification through synthetic experiments in the next section.

As we will show in the following subsection, we initially break the conditions of learning the spurious feature by letting $\hat{\alpha} = 1/2$ in a group-balanced data subset. Subsequently, we utilize the momentum to amplify the core feature's coefficient, allowing for tolerance of $\hat{\alpha} > 1/2$ when adding new data.

6.2.2 PDE: A Two-Stage Training Algorithm

We present a new algorithm named Progressive Data Expansion (PDE) in Algorithm 5 and explain the details below, which consist of (1) warm-up and (2) expansion stages.

As accelerated gradient methods are most commonly used in applications, we jointly consider the property of momentum and our theoretical insights when designing the algorithm. For gradient descent with momentum (GD+M), at each iteration t and with momentum coefficient $\gamma > 0$, it updates as follows

$$\mathbf{g}^{(t+1)} = \gamma \mathbf{g}^{(t)} + (1-\gamma)\nabla \mathcal{L}(\mathbf{W}^{(t)}), \tag{6.6}$$

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \cdot \mathbf{g}^{(t+1)}, \tag{6.7}$$

Warm-up Stage. In this stage, we create a fully balanced dataset S^0, in which each group is randomly subsampled to match the size of the smallest group, and consider it as a

Algorithm 5 Progressive Data Expansion (PDE)

```
Require: Number of iterations $T_0$ for warm-up training; number of times $K$ for dataset expansion; number of iterations $J$ for expansion training; number of data $m$ for each expansion; learning rate $\eta$; momentum coefficient $\gamma$; initialization scale $\sigma_0$; training set $S = \{(\mathbf{x}_i, y_i, a_i)\}_{i=1}^n$; model $f_{\mathbf{W}}$.
Initialize $\mathbf{W}^{(0)}$.
Warm-up stage
Divide the $S$ into groups by values of $y$ and $a$: $S_{y,a} = \{(\mathbf{x}_i, y_i, a_i)\}_{y_i=y, a_i=a}$.
Generate warm-up set $S^0$ from $S$ by randomly subsampling from each group of $S$ such that $|S^0_{y,a}| = \min_{y',a'} |S_{y',a'}|$ for $y \in \{\pm 1\}$ and $a \in \{\pm 1\}$.
for $t = 0, 1, \ldots, T_0$ do
    Compute loss on $S^0$: $\mathcal{L}_{S^0}(\mathbf{W}^{(t)}) = \frac{1}{|S^0|} \sum_{i \in S^0} \ell(y_i f(x_i; \mathbf{W}^{(t)}))$.
    Update $\mathbf{W}^{(t+1)}$ by (6.6) and (6.7).
end for
Expansion stage
for $k = 1, \ldots, K$ do
    Draw $m$ examples ($S_{[m]}$) from $S/S^{k-1}$ and let $S^k = S^{k-1} \cup S_{[m]}$.
    for $t = 1, \ldots, J$ do
        Compute loss on $S^k$: $\mathcal{L}_{S^k}(\mathbf{W}^{(T)}) = \frac{1}{|S^k|} \sum_{i \in S^k} \ell(y_i f(x_i; \mathbf{W}^{(T)}))$, where $T = T_0 + (k-1) * J + t$.
        Update $\mathbf{W}^{(T+1)}$ by (6.6) and (6.7).
    end for
end for
return $\mathbf{W}^{(t)} = \arg\max_{\mathbf{W}^{(t')}} \mathrm{Acc}^{\mathrm{val}}_{\mathrm{wg}}(\mathbf{W}^{(t')})$.
```

warm-up dataset. We train the model on the warm-up dataset for a fixed number of epochs. During this phase, the model is anticipated to accurately learn the core feature without being influenced by the spurious feature. Note that, under our data model, a completely balanced dataset will have $\hat{\alpha} = 1/2$. We present the following lemma as a theoretical basis for the warm-up stage.

Lemma 6.2.1. *Given the balanced training dataset $S^0 = \{(\mathbf{x}_i, y_i, a_i)\}_{i=1}^{N_0}$ with $\hat{\alpha} = 1/2$ as in Definition 6.1.1 and CNN as in (6.3). The gradient on $\boldsymbol{v}_s$ will be 0 from the beginning of training.*

In particular, with $\hat{\alpha} = 1/2$ we have $|S_1^0| = |S_2^0|$: an equal amount of data is positively correlated with the spurious feature as the data negatively correlated with the spurious feature. In each update, both groups contribute nearly the same amount of spurious feature gradient with different signs, resulting in cancellation. Ultimately, this prevents the model from learning the spurious feature. Detailed proofs can be found in Appendix 9.5.6.

Expansion Stage. In this stage, we proceed to train the model by incrementally incorporating new data into the training dataset. The rationale for this stage is grounded in the theoretical result by the previous work [81] on GD with momentum, which demonstrates that once gradient descent with momentum initially increases its correlation with a feature $\boldsymbol{v}$, it retains a substantial historical gradient in the momentum containing $\boldsymbol{v}$. Put it briefly, the initial learning phase has a considerable influence on subsequent training for widely-used accelerated training algorithms. While ERM learns the spurious feature $\boldsymbol{v}_s$ and momentum does not help, as we will show in synthetic experiments, PDE avoids learning $\boldsymbol{v}_s$ and learns $\boldsymbol{v}_c$ in the warm-up stage. This momentum from warm-up, in turn, amplifies the core feature that is present in the gradients of newly added data, facilitating the continued learning of $\boldsymbol{v}_c$ in the expansion stage. For a specific illustration, the learning of the core feature by GD+M will be

$$\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c \rangle = \langle \mathbf{w}_j^{(t)} - \eta\big(\gamma g^{(t)} + (1-\gamma)\nabla_{\mathbf{w}_j}\mathcal{L}(\mathbf{W}^{(t)})\big), \boldsymbol{v}_c \rangle,$$

where $g^{(t)}$ is the additional momentum as compared to GD with $\gamma = 0$. While the current gradient along $\boldsymbol{v}_c$ might be small (i.e., β_c), we can benefit from the historical gradient in $g^{(t)}$ to amplify the growth of $\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c \rangle$ and make it larger than that of the spurious feature (i.e., β_s). This learning process will then correspond to the case when the model learns the core feature discussed in Subsection 6.2.1. Practically, we consider randomly selecting m new examples for expansion every J epochs by attempting to draw a similar number of examples from each group. During the last few epochs of the expansion stage, we expect the newly incorporated data exclusively from the larger group, as the smaller groups have been entirely integrated into the warm-up dataset.

It is worth noting that while many works address the issue of identifying groups from datasets containing spurious correlations, we assume the group information is known and our algorithm focuses on the crucial subsequent question of optimizing group information utilization. Aiming to prevent the learning of spurious features, PDE distinguishes itself by employing a rapid and lightweight warm-up stage and ensuring continuous improvement during the expansion stage with the momentum acquired from the warm-up dataset. Our training framework is both concise and effective, resulting in computational efficiency and ease of implementation.

6.3 Experiments

In this section, we present the experiment results from both synthetic and real datasets. Notably, we report the worst-group accuracy, which assesses the minimum accuracy across all groups and is commonly used to evaluate the model's robustness against spurious correlations.

6.3.1 Synthetic Data

In this section, we present synthetic experiment results in verification of our theoretical findings. In Appendix 9.5.1, we illustrate the detailed data distribution, hyper-parameters of the experiments and more extensive experiment results. The data used in this section is

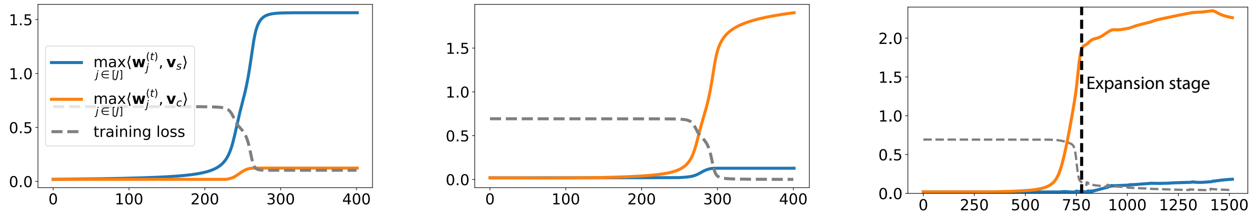
Table 6.1: Synthetic data experiments. We report the worst-group accuracy and the gap (i.e., overall - worst). We further consider several variations of PDE to demonstrate the importance of each component of our method. **Reset**: we reset the momentum to zero after the warm-up stage. **Warmup+All**: we let PDE incorporate all of the new training data at once after the warm-up stage.

	Worst-group (%)	Gap (%)
ERM (GD)	0.00	97.71
ERM (GD+M)	0.00	97.71
Warmup+All (Reset)	67.69	31.18
Warmup+All	74.24	24.76
PDE (Reset)	92.51	2.29
PDE	**93.01**	**0.03**

generated following Definition 6.1.1. We consider the worst-group and overall test accuracy. As illustrated in Table 6.1, ERM, whether trained with GD or GD+M, is unable to accurately predict the small group in our specified data distribution where $\hat{\alpha} = 0.98$ and $\beta_c < \beta_s$. In contrast, our method significantly improves worst-group accuracy while maintaining overall test accuracy comparable to ERM. Furthermore, as depicted in Figure 6.3a, ERM rapidly learns the spurious feature as it minimizes the training loss, while barely learning the core feature. Meanwhile, in Figure 6.3b we show the learning of ERM when the data distribution breaks the conditions of our theory and has $\beta_c > \beta_s$ instead. Even with the same $\hat{\alpha}$ as in Figure 6.3a, ERM correctly learns the core feature despite the imbalanced group size. These two figures support the theoretical results we discussed to motivate our method. Consequently, on the same training dataset as in Figure 6.3a, Figure 6.3c shows that our approach allows the model to initially learn the core feature using the warm-up dataset and continue learning when incorporating new data.

6.3.2 Real Data

We conduct experiments on real benchmark datasets to (1) compare our approach with state-of-the-art methods, highlighting its superior performance and efficiency, and (2) offer insights into the design of our method through ablation studies.



(a) ERM (case 1)

(b) ERM (case 2)

(c) PDE

Figure 6.3: Comparison of methods in different scenarios.

Figure 6.4: **Training process of ERM vs. PDE.** We consider the same dataset generated from the distribution as in Definition 6.1.1 for ERM (case 1) and PDE. On the same training data, ERM learns the spurious feature while PDE successfully learns the core feature. We further consider ERM (case 2) when training on the data distribution where $\beta_c > \beta_s$ and $\hat{\alpha} = 0.98$. We show the growth of the max inner product between the model's neuron and core/spurious signal vector and the decrease of training loss with regard to the number of iterations t.

Datasets. We evaluate on three wildly used datasets across vision and language tasks for spurious correlation: (1) **Waterbirds** [155] contains bird images labeled as waterbird or landbird, placed against a water or land background, where the smallest subgroup is waterbirds on land background. (2) **CelebA** [109] is used to study gender as the spurious feature for hair color classification, and the smallest group in this task is blond-haired males. (3) **CivilComments-WILDS** [93] classifies toxic and non-toxic online comments while dealing with demographic information. It creates 16 overlapping groups for each of the 8 demographic identities.

Baselines. We compare our proposed algorithm against several state-of-the-art methods. Apart from standard ERM, we include GroupDRO [155] and DFR [90] that assume access to the group labels. We note that DFR$^{\text{Val}}$ also uses the validation data for fine-tuning the last layer of the model. We also design a baseline called subsample that simply trains the model on the warm-up dataset only. Additionally, we evaluate three recent methods that address spurious correlations without the need for group labels: LfF [132], EIIL [43], and JTT [106]. We report results for ERM, Subsample, GroupDRO and PDE based on our own runs using the WILDS library [92]; for others, we directly reuse their reported numbers.

We present the experiment details including dataset statistics and hyperparameters as

Table 6.2: The worst-group and average accuracy (%) of PDE compared with state-of-the-art methods. The **bold** numbers indicate the best results among the methods that *require group information*, while the underscored numbers represent methods that *only train once*. All methods use validation data for early stopping and model selection, while √√ indicates that the method also re-trains the last layer using the validation data

Method	**Group info**	Train once	Val info	Waterbirds		CelebA		CivilComments	
				Worst	Average	Worst	Average	Worst	Average
ERM	×	√	√	$70.0_{\pm 2.3}$	$97.1_{\pm 0.1}$	$45.0_{\pm 1.5}$	$94.8_{\pm 0.2}$	$58.2_{\pm 2.8}$	$92.2_{\pm 0.1}$
LfF	×	×	√	$78.0_{N/A}$	$91.2_{N/A}$	$77.2_{N/A}$	$85.1_{N/A}$	$58.8_{N/A}$	$92.5_{N/A}$
EIIL	×	×	√	$77.2_{\pm 1.0}$	$96.5_{\pm 0.2}$	$81.7_{\pm 0.8}$	$85.7_{\pm 0.1}$	$67.0_{\pm 2.4}$	$90.5_{\pm 0.2}$
JTT	×	×	√	$86.7_{N/A}$	$93.3_{N/A}$	$81.1_{N/A}$	$88.0_{N/A}$	$69.3_{N/A}$	$91.1_{N/A}$
Subsample	√	√	√	$86.9_{\pm 2.3}$	$89.2_{\pm 1.2}$	$86.1_{\pm 1.9}$	$91.3_{\pm 0.2}$	$64.7_{\pm 7.8}$	$83.7_{\pm 3.4}$
DFR$^{\text{Tr}}$	√	×	√	$90.2_{\pm 0.8}$	$97.0_{\pm 0.3}$	$80.7_{\pm 2.4}$	$90.6_{\pm 0.7}$	$58.0_{\pm 1.3}$	$92.0_{\pm 0.1}$
DFR$^{\text{Val}}$	√	×	√√	**92.9**$_{\pm 0.2}$	$94.2_{\pm 0.4}$	$88.3_{\pm 1.1}$	$91.3_{\pm 0.3}$	$70.1_{\pm 0.8}$	$87.2_{\pm 0.3}$
GroupDRO	√	√	√	$86.7_{\pm 0.6}$	$93.2_{\pm 0.5}$	$86.3_{\pm 1.1}$	$92.9_{\pm 0.3}$	$69.4_{\pm 0.9}$	$89.6_{\pm 0.5}$
PDE	√	√	√	$90.3_{\pm 0.3}$	$92.4_{\pm 0.8}$	**91.0**$_{\pm 0.4}$	$92.0_{\pm 0.6}$	**71.5**$_{\pm 0.5}$	$86.3_{\pm 1.7}$

Table 6.3: Training efficiency of PDE and GroupDRO on Waterbirds. We compare with GroupDRO at their learning rate and weight decay, as well as at ours. We report the worst-group accuracy, average accuracy and the number of epochs till early stopping as the model reached the best performance on validation data. Note: for a fair comparison, we consider one training epoch as training over the N data as the size of the training dataset.

Method	Learning rate	Weight decay	Worst	Average	Early-stopping epoch*
GroupDRO	1e-5	1e-0	$86.7_{\pm 0.6}$	$93.2_{\pm 0.5}$	$92_{\pm 4}$
GroupDRO	1e-2	1e-2	$77.3_{\pm 2.0}$	$97.1_{\pm 0.5}$	$15_{\pm 15}$
PDE	1e-2	1e-2	**90.3**$_{\pm 0.3}$	$92.4_{\pm 0.8}$	$8.9_{\pm 1.8}$

well as comprehensive additional experiments in Appendix 9.5.2.

6.3.2.1 Consistent Superior Worst-group Performance

We assess PDE on the mentioned datasets with state-of-the-art methods. Importantly, we emphasize the comparison with GroupDRO, as it represents the best-performing method that utilizes group information. As shown in Table 6.2, PDE considerably enhances the worst-performing group's performance across all datasets, while maintaining the average accuracy comparable to GroupDRO. Considering all methods that only use validation data for model selection, GroupDRO still occasionally fails to surpass other methods. Remarkably PDE's performance consistently exceeds them in worst-group accuracy.

Table 6.4: Performance of PDE after each stage. We report the worst-group and average accuracy.

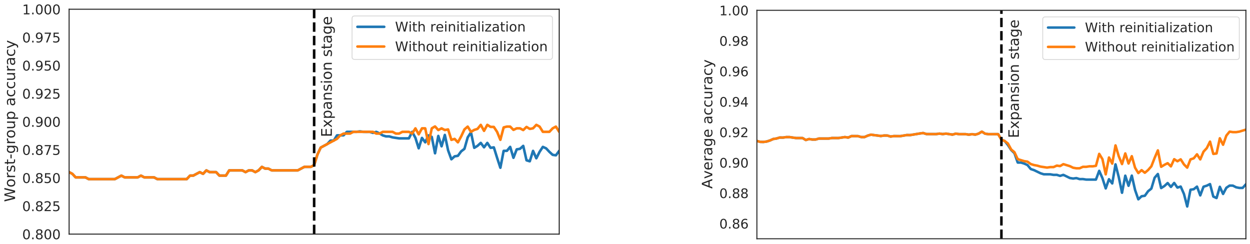
	Warm-up		Addition	
Dataset	Worst	Avg	Worst	Avg
Waterbirds	86.0	91.9	**90.3**	92.4
CelebA	87.8	92.1	**91.0**	92.0
CivilComm	67.7	78.8	**71.5**	86.3

6.3.2.2 Efficient Training

In this subsection, we show that our method is more efficient as it does not train a model twice (as in JTT) and more importantly avoids the necessity for a small learning rate (as in GroupDRO). Specifically, methods employing group-balanced batches like GroupDRO require a very small learning rate coupled with a large weight decay in practice. We provide an intuitive explanation as follows. When sampling to achieve balanced groups in each batch, smaller groups appear more frequently than larger ones. If training progresses rapidly, the loss on smaller groups will be minimized quickly, while the majority of the large group data remains unseen and contributes to most of the gradients in later batches. Therefore, these methods necessitate slow training to ensure the model encounters diverse data from larger groups before completely learning the smaller groups. We validate this observation in Table 6.3, where GroupDRO trained faster than the default results in significantly poorer performance similar to ERM. Conversely, PDE can be trained to converge rapidly on the warm-up set and reaches better worst-group accuracy 10× faster than GroupDRO at default. Note that methods which only finetune the last layer [90, 200] are also efficient. However, they still require training a model first using ERM on the entire training data till convergence. In contrast, PDE does not require further finetuning of the model.

6.3.2.3 Understanding the Two Stages

We examine each component and demonstrate their effect in PDE. In Table 6.4, we present the worst-group and average accuracy of the model trained following the warm-up and expansion stages. Indeed, the majority of learning occurs in the warm-up stage, during which a satisfactory worst-group accuracy is established. In the expansion stage, the model persists in learning new data along the established trajectory, leading to continued performance improvement. In Figure 6.6, we corroborate and emphasize that the model has acquired the appropriate features and maintains learning based on its historical gradient stored in the momentum. As shown, if the optimizer is reset after the warm-up stage and loses all its historical gradients (with reinitialization), it soon acquires spurious features, resulting in a swift decline in performance accuracy as shown in the blue line.



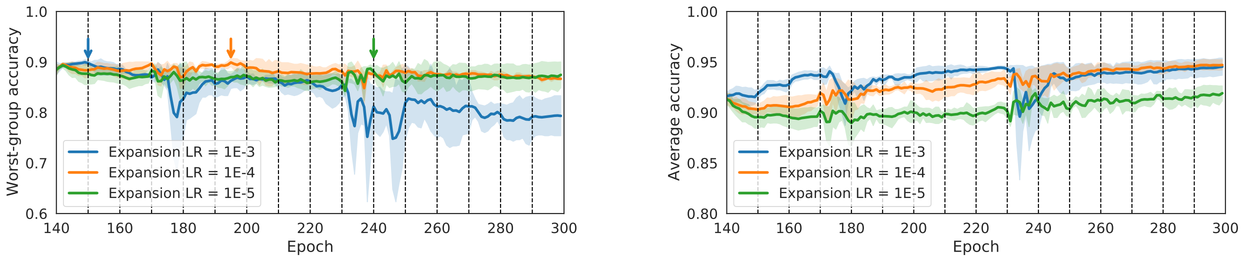
(a) Worst-group accuracy.

(b) Average accuracy.

Figure 6.5: The effect of resetting the momentum after the warm-up stage for PDE on Waterbirds.

6.3.2.4 Ablation Study on the Hyper-parameters of PDE

PDE is robust within a reasonable range of hyperparameter choices, although some configurations outperform others. As shown in Table 6.5, it is necessary to limit the number of data points introduced during each expansion to prevent performance degradation. Similarly, in Appendix 9.5.1, we emphasize the importance of gradual data expansion. In Table 6.6, we show that post-warmup learning rate decay is essential, though PDE exhibits tolerance to the degree of this decay. Lastly, as illustrated in Figure 6.6, adopting a smaller learning rate often necessitates increased data expansions. Nonetheless, a reduced learning rate does not necessarily lead to improved performance.



(a) Worst-group accuracy.

(b) Average accuracy.

Figure 6.6: The variations in both worst-group and average accuracy on the test set of Waterbirds during the expansion stage under different expansion learning rates. Each vertical dashed line denotes an expansion and the arrow denotes the early stopping.

Table 6.5: Ablation study on Waterbirds. Exp. size: number of data points added in each expansion.

Exp. size	Exp. lr	Worst	Average
5	1e-4	89.9$_{\pm 0.5}$	92.1$_{\pm 0.3}$
10	1e-4	**90.3**$_{\pm 0.3}$	92.4$_{\pm 0.8}$
50	1e-4	88.1$_{\pm 0.8}$	93.4$_{\pm 0.4}$

Table 6.6: Ablation study on Waterbirds. Exp. lr: the learning rate in the expansion stage.

Exp. size	Exp. lr	Worst	Average
10	1e-2	85.4$_{\pm 3.1}$	92.1$_{\pm 2.0}$
10	1e-3	89.4$_{\pm 0.7}$	92.6$_{\pm 0.3}$
10	1e-5	89.5$_{\pm 0.2}$	92.1$_{\pm 0.1}$

6.4 Related Work

Existing approaches for improving robustness against spurious correlations can be categorized into two lines of research based on the tackled subproblems. A line of research focuses on the same subproblem we tackle: effectively using the group information to improve robustness. With group information, one can use the distributionally robust optimization (DRO) framework and dynamically increase the weight of the worst-group loss in minimization [72, 139, 155, 227]. Within this line of work, GroupDRO [155] achieves state-of-the-art performances across multiple benchmarks. Other approaches use importance weighting to reweight the groups [166, 27, 208] and class balancing to downsample the majority or upsample the minority [66, 45, 156]. Alternatively, **(author?)** [62] leverage group information to augment the minority groups with synthetic examples generated using GAN. Another strategy [28, 29] involves imposing Lipschitz regularization around minority data points. Most recently, methods that train a model using ERM first and then only finetune the last layer on balanced data from training or validation [90], or on mixed representations [210], or learn post-doc scaling adjustments [200] are shown to be effective.

The other line of research focuses on the setting where group information is not available during training and tackles the first subproblem we identified as accurately finding the groups. Recent notable works [132, 106, 43, 229, 217] mostly involve training two models, one of which is used to find group information. To finally use the found groups, many approaches [135, 55, 139, 172] still follow the DRO framework.

The first theoretical analysis of spurious correlation is provided by **(author?)** [156]. For self-supervised learning, **(author?)** [35] shows that fine-tuning with pre-trained models can reduce the harmful effects of spurious features. **(author?)** [221] provides guarantees in the presence of label noise that core features are learned well only when less noisy than spurious features. These theoretical works only provide analyses of linear models. Meanwhile, a parallel line of work has established theoretical analysis of nonlinear CNNs in the more realistic setting **(author?)** [10, 239, 201, 37, 81]. Our work builds on this line of research and generalizes it to the study of spurious features. Lastly, we notice that a concurrent work [36] also uses tensor power method [10] to analyze the learning of spurious features v.s. invariant features, but in the setting of out-of-distribution generalization.

6.5 Conclusion

In conclusion, this paper addressed the challenge of spurious correlations in training deep learning models and focused on the most effective use of group information to improve robustness. We provided a theoretical analysis based on a simplified data model and a two-layer nonlinear CNN. Building upon this understanding, we proposed PDE, a novel training algorithm that effectively and efficiently enhances model robustness against spurious correlations. This work contributes to both the theoretical understanding and practical application of mitigating spurious correlations, paving the way for more reliable and robust deep learning models.

Limitations and future work. Although beyond the linear setting, our analysis still focuses on a relatively simplified binary classification data model. To better represent real-world application scenarios, future work could involve extending to multi-class classification problems and examining the training of transformer architectures. Practically, our proposed method requires the tuning of additional hyperparameters, including the number of warm-up epochs, the number of times for dataset expansion and the number of data to be added in each expansion.

CHAPTER 7

Fine-tuning against Spurious Correlations for Vision-Language Models

Vision-Language models (e.g., CLIP, DALL-E, Stable Diffusion, Imagen) are becoming pervasive in real-world deployments and have transformed the way large-scale model architectures are trained and used in different applications. Their multi-modal nature has not only enabled a large variety of tasks (e.g. text-to-image generation, visual question answering, image captioning) but is also facilitating better learning techniques that take advantage of data in several modalities to jointly learn embeddings that can then be reused in downstream tasks [150, 86, 99, 237].

While the multi-modal alignment increases the expectations about model reliability due to better grounding and larger availability of data in general, these models are still not immune to fundamental learning problems such as dealing with spurious correlations [21, 129, 145, 4]. Therefore, when such models are used as a backbone to solve application-oriented tasks on a given domain, existing spurious correlations specific to that domain or the fine-tuning data that comes with it, may resurface in ways that are harmful to end users. At the same time, retraining large models from scratch to address such issues has become a less realistic avenue for two main reasons. First, stakeholders who need to adapt a model to a particular domain may not necessarily have access to large-scale computation. Second, the types of spurious correlations of interest are often domain-specific and not all of them can be anticipated ahead of time during pre-training of a general model. Furthermore, while previous work has studied spurious correlations in single-modal models trained with supervised learning, we note that spurious correlations learned in a joint multimodal embedding space with contrastive

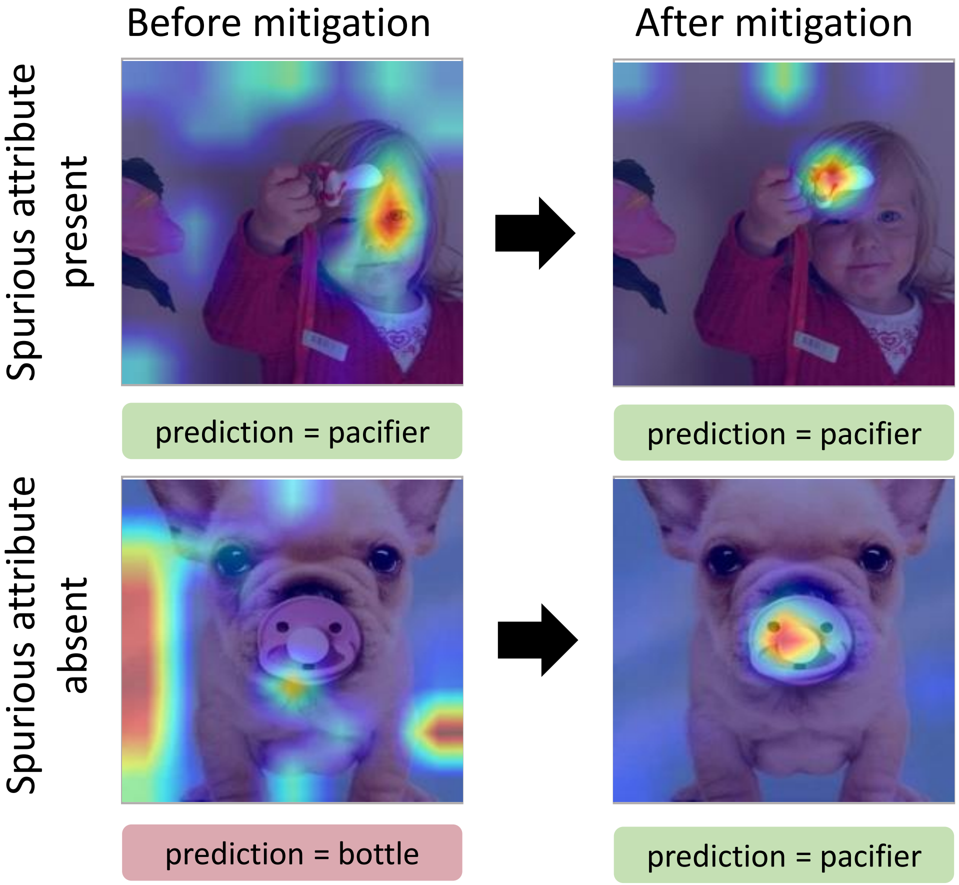


Figure 7.1: The baby pacifier class in ImageNet is spuriously correlated with the presence of babies, which leads the pre-trained model to be less accurate for cases when babies are absent in the image (bottom row) and also be right for the wrong reasons when babies are present (top row). Our approach mitigates both concerns by conveniently expressing and decorrelating the spurious relationships in the loss function via language.

language image pretraining may not be the same due to differences in inputs and training objectives. For instance, we found that certain spurious correlations commonly studied in supervised learning of vision models, such as the correlation between gender and hair colors in the CelebA dataset [109], were not learned by multimodal models with contrastive language image pretraining. This suggests that spurious correlations in multimodal models may exhibit unique characteristics that require further investigation.

Building on the challenges of spurious correlations in vision-language models and the need for efficient mitigation methods, we introduce a contrastive learning approach that leverages the multi-modality of CLIP as a vision-language model to detect and mitigate spurious correlations through language in fine-tuning time. In the detection stage, our method extracts linguistic attributes from the image and tests whether their presence or absence affects model performance. If the accuracy of the model drops when a specific attribute is not present, it indicates that the attribute is either an overemphasized but necessary attribute (e.g., misclassifying taxi cabs that are not yellow) or a spurious correlation (e.g., misclassifying boats when there is no water in the background) [171]. Assuming that a practitioner or domain expert in the loop can determine whether the attribute is healthy or spurious, in the next stage, our method mitigates the identified spurious correlation by extending the current contrastive language-vision learning techniques with a set of additional loss functions that explicitly i) decorrelate spurious attributes from the class names in language, and ii) push away both the vision representations across classes and language representations of templates substituted with different class labels. It is worth noting that our approach only fine-tunes the projections to the joint embedding space. Since the projection layers contain much fewer parameters than the full models, our method requires significantly less computational resources compared to extensive retraining from scratch without losing features learned in pretraining.

In contrast to previous work which requires human annotations about spurious or group attributes [155], our approach uses automatically detected language-based descriptions of spurious attributes that can then directly be expressed and used in optimization to set them apart from affected classes. While domain experts are still required in this method to judge whether a detected co-occurence is a spurious attribute or not, this still minimizes labeling human supervision per example. Fine-tuning experiments with two datasets, Waterbirds and Imagenet, show that the proposed approach offers a better trade off between the average accuracy and worst-group accuracy (i.e., examples when the spurious attribute is not present) and can better align model explanation maps to the class of interest.

It is worth noting that our work differs from existing studies that focus on spurious correlations learned by vision models [155, 132, 43, 106, 134, 77]. Instead, we investigate spurious correlations learned by multimodal models during pre-training with the contrastive language-image loss. Although larger models may be less accurate than specialized models on certain tasks, practitioners may still choose to use a pretrained model for reasons such as maintenance and data availability. In addition, having enough labeled data to train a specialized vision model may not always be possible. In such cases, the larger pretrained model trained on noisy image-caption pairs may have already encoded useful information about the concept, and our method is useful for scenarios where one needs to maintain this generality while mitigating found issues for a specific domain.

Moreover, the multimodal nature of these models opens up new opportunities for detecting and mitigating failures without the need for additional annotation data, such as attributes or bounding boxes, to guide the model's attention. By leveraging the information encoded in the joint embedding space, our approach improves the model's attention in GradCAM explanations and quantitatively in AIoU scores, a new metric we proposed for evaluating the model's attention. This finding is particularly noteworthy as the need for metadata annotations and grounding has been a significant barrier for several applications, especially during cold starts.

In summary, our contributions are:

- A language-based approach that detects spurious correlations with practitioner supervision but no spurious attribute labeling.
- A loss function that extends current contrastive vision-language learning for mitigating spurious correlations in vision through language.
- A set of experiments that showcase how to use the proposed detection and mitigation approach in practice for the CLIP model as well as its effectiveness in datasets with known and unknown spurious correlations.

7.1 Related Work

Explaining and Debugging Trained Models. Several algorithms have been proposed to semantically explain and analyze trained neural networks, including distilling the decision modes into decision trees [230, 171], training classifiers in the latent space [78, 212], and embedding inputs with joint vision-language representations to find the error slices with a mixture model [57]. These methods usually accompany the semantic explanations with feature attention maps, e.g., GradCam [162]. The authors of [165] conducted a comprehensive study on ImageNet [153] by manually relabeling it and uncovered multiple instances of label noise and disagreement in the dataset. In this paper, we are only interested in discovering and mitigating *spurious correlations*, which are introduced next.

Enhancing Robustness to Spurious Correlations. We study spurious correlations in the context of deep learning, as they have been formally discussed in [155]. Given a classification dataset $\mathcal{D}$ with labels $\mathcal{Y}$, if there exist spurious attributes $\mathcal{A}$ that are highly correlated with $\mathcal{Y}$, a deep neural network trained on this dataset is likely to learn $\mathcal{A}$ as features to distinguish $\mathcal{Y}$, even if the attribute is not conceptually part of the class concept. For example, in Figure 7.1, a pretrained CLIP-RN50 model [150] learned to use *baby* to identify *baby pacifier* because they often appear together in ImageNet [153], instead of actually learning the baby pacifier itself.

To prevent deep learning models from learning such spurious correlations from biased data, recent work proposed training strategies robust to spurious attributes for either vision or language models [155, 132, 43, 106, 134, 77]. The spurious label of each training example (e.g., whether this example contains the spurious feature) is either provided [155, 77] or inferred by training a reference model [132, 43, 106, 134] until it learns the spurious correlations. Other approaches indirectly estimate and use the causal effect of hidden non-labeled spurious attributes in pre-training [117].

However, these studies all focus on training *unimodal* models with datasets that contain known spurious features, and spurious correlations learned by pretrained *multimodal* models have not been extensively studied. To the best of our knowledge, we are the first to propose a *fine-tuning* approach for mitigating spurious correlations in multimodal models. While [228] also studied CLIP's robustness to group shifts including spurious correlations, their method is designed for transfer learning rather than fine-tuning the learned embedding space.

Correcting Vision Models using Language. There is a line of recent work aiming to fix vision classifiers with language inputs. **(author?)** [145] uses attention maps from a pre-trained CLIP to supervise a CNN classifier's spatial attention. **(author?)** [232] probes a vision classifier trained on the joint vision-language embedding space of CLIP using language embeddings of attributes, identifies the attributes causing most failures, and generates a large set of natural language inputs with the influential attributes to rectify the model. However, this line of work aims to guide CNN classifiers rather than fixing CLIP models and does not prevent spurious feature usage.

7.2 Spurious-aware Contrastive Language Image Fine-tuning

Background. Contrastive Language-Image Pretraining (CLIP) learns from millions of image caption pairs, by maximizing the agreement between representations of every image and the representations of its corresponding caption. Specifically, the CLIP architecture consists of (i) an image encoder network, (ii) a text encoder network, and (iii) a contrastive objective that pulls the embeddings of every image and its corresponding caption together while pushing apart embeddings of the image from other captions in the same minibatch. Formally, for a minibatch of N image-captions pairs $\{I_j, T_j\}_{j=1}^N$, and their encoded embeddings $\{I_j^e, T_j^e\}_{j=1}^N$, the CLIP loss is defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{CLIP}} = &-\frac{1}{2}\mathbb{E}_{(I_i,T_i)} \log \left[\frac{e^{\left\langle I_j^e, T_j^e\right\rangle/\tau}}{\sum_{k=1}^N e^{\left\langle I_j^e, T_k^e\right\rangle/\tau}}\right] \\ &-\frac{1}{2}\mathbb{E}_{(I_i,T_i)} \log \left[\frac{e^{\left\langle I_k^e, T_k^e\right\rangle/\tau}}{\sum_{j=1}^N e^{\left\langle I_j^e, T_k^e\right\rangle/\tau}}\right],\end{aligned} \tag{7.1}$$

where $\langle ., . \rangle$ represents the inner product, and τ is a trainable temperature parameter. For finetuning CLIP on a dataset of images and their labels, such as Waterbirds, the labels are replaced in the engineered prompt templates, such as “A photo of a {label}", “A photo of a {label}, a type of bird.”, etc. Then, the loss is minimized on the images paired with templates built with image labels. We use all 80 templates described in [150].

For a given spurious attribute (e.g. water or land background in the Waterbirds dataset), we will use the following losses to eliminate the spurious correlation during fine-tuning. Please note that the contrastive losses below use the class information to pull together representations of examples from the same class label, and push away representations of examples from different class labels. The spurious losses use the spurious attribute detected in the spurious correlation detection stage (Section 7.3) to pull together representations of examples with the same spurious attribute (e.g. attribute present) and push away representations of examples with a different spurious attribute (e.g. attribute absent).

Here, we will use the following construct as a basis for the definition of all loss terms: a cross-group representation similarity term that pulls together representations from the same group and pushes away representations of different groups. The representations can be either in the vision or language space. We reuse this construct to extend CLIP contrastive learning to improve classification and also mitigate spurious correlations. Let $G_1 = \{(I_p, T_p)\}_{p=1}^{P}$ be the set of examples in one group of examples in the minibatch, and $G_2 = \{(I_q, T_q)\}_{q=1}^{Q}$ the set of examples in another group of the same minibatch, as defined by the relationship of a given example in the minibatch (I_i, T_i) to these groups. Depending on the loss term, the relationship between examples can be either due to examples belonging to the same class or having the same spurious attribute value. Then, the cross-group representation similarity defined across two modalities of representation embeddings A and B is:

$$\mathrm{CS} = \mathbb{E}\,{-}_{\substack{(I_i,T_i),\\ (I_p,T_p)\in G_1\\ (I_q,T_q)\in G_2}} \left[\log \frac{e^{\left\langle A_i^e, B_p^e \right\rangle/\tau}}{\sum_{p=1}^{P} e^{\left\langle A_i^e, B_p^e \right\rangle/\tau} + \sum_{q=1}^{Q} e^{\left\langle A_i^e, B_q^e \right\rangle/\tau}} \right]$$

Contrastive Image Loss The first term is a contrastive image loss which pulls together image representations of a class, and pushes away image representations of different classes in the vision model. Let $G_l = \{(I_p, T_p)\}_{p=1}^{P}$ be the set of examples in the minibatch with the **same label** as example (I_i, T_i), i.e., $T_i = T_p$, and $\hat{G}_l = \{(I_q, T_q)\}_{q=1}^{Q}$ be the set of examples with a **different label**. Then the contrastive image loss within the vision representation embeddings I is defined as:

$$\mathcal{L}_{vc} = \mathrm{CS}(G_l, \hat{G}_l, I, I) \tag{7.2}$$

Contrastive Language Loss The second term is a contrastive language loss which pulls together language representations of templates of a class in the language model, and pushes away language representations of different classes. Let $G_l = \{(I_p, T_p)\}_{p=1}^{P}$ be the set of examples in the minibatch with the **same label** as example (I_i, T_i), i.e., $T_i = T_p$, but with different templates. Let $\hat{G}_l = \{(I_q, T_q)\}_{q=1}^{Q}$ be the set of examples in the minibatch with a **different label**. Then the contrastive language loss within the language representation embeddings T is defined as:

$$\mathcal{L}_{lc} = \mathrm{CS}(G_l, \hat{G}_l, T, T) \tag{7.3}$$

Spurious Image Loss The third term is a spurious contrastive image loss which pulls together image representations of each group of examples in a class, and pushes away image representations of different groups of examples. For example, it pulls together images of waterbirds with water background, and pulls them away from images of waterbirds with land background and from landbird images with water or land background.

Assume $G_s = \{(I_p, T_p)\}_{p=1}^{P}$ is the set of images in the minibatch with the **same spurious attribute** as example (I_i, T_i), and $\{\hat{G}_s = (I_q, T_q)\}_{q=1}^{Q}$ is the set of examples with a **different spurious attribute** than example (I_i, T_i). A different spurious attribute here could also mean that the spurious attribute is absent. Then, the spurious image loss within the vision representation embeddings I is defined as:

$$\mathcal{L}_{vs} = \mathrm{CS}(G_s, \hat{G}_s, I, I) \tag{7.4}$$

Spurious Language Loss The last term is a spurious contrastive language loss which pulls together language representations of each group of examples in a class, and pushes away language representations of different groups of examples. Assume $G_s = \{(I_p, T_p)\}_{p=1}^{P}$ is the set of examples in the minibatch with the **same spurious attribute** with example (I_i, T_i), and $\hat{G}_s = \{(I_q, T_q)\}_{q=1}^{Q}$ is the set of examples with a **different spurious attribute** in the minibatch. Note that, a different spurious attribute here could also mean that a spurious attribute is absent. Then, the spurious language loss within the language representation embeddings T is defined as:

$$\mathcal{L}_{ls} = \mathrm{CS}(G_s, \hat{G}_s, T, T) \tag{7.5}$$

The final loss is the sum of all the terms above:

$$\mathcal{L} = \mathcal{L}_{\mathrm{CLIP}} + \mathcal{L}_{vc} + \mathcal{L}_{lc} + \mathcal{L}_{vs} + \mathcal{L}_{ls}. \tag{7.6}$$

In practice, either $\mathcal{L}_{vs}$ or $\mathcal{L}_{ls}$ can be combined with $\mathcal{L}_{lc}$ to effectively eliminate the spurious correlation. If spurious attribute annotation labels are available, one can use $\mathcal{L}_{vs}$. If spurious attribute annotation labels are not available $\mathcal{L}_{ls}$ can provide a good separation between groups in different classes. In all experiments reported hereafter we show results for both, and the ablation study in Section 9.6.1 details the tradeoffs between these and other choices.

From an implementation perspective, all language-related losses could be implemented across examples or templates. Our implementation follows a template-based approach.

7.3 Spurious Correlation Detection

This section introduces our pipeline for detecting and evaluating spurious correlations learned by a pretrained model. While we apply this pipeline to CLIP models in this work, it can be generalized to other pretrained models as well. The approach closely follows previously discussed techniques [171, 138] but relies on automatically generated annotations for attributes.

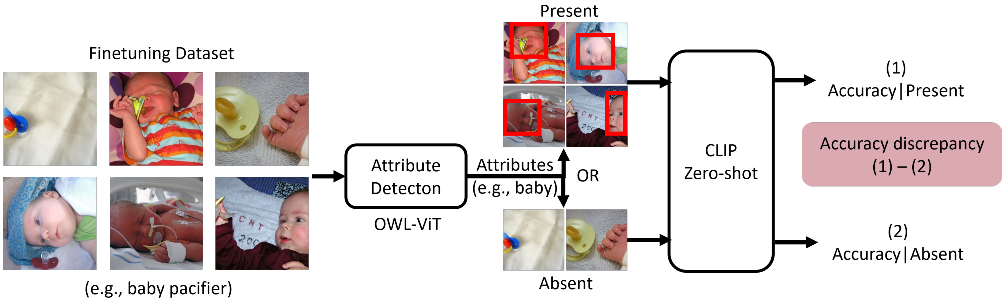


Figure 7.2: Spurious correlation detection based on attributes from an open-vocabulary detector and accuracy discrepancy scores of the model between examples when the spurious attribute is present or absent.

7.3.1 Methodology

For any given fine-tuning dataset, we are interested in knowing whether CLIP (or any other pretrained models) has learned any spurious correlations for the classes in the dataset. According to the definition of spurious attributes introduced in Section 7.2, models that have learned a certain spurious correlation usually show better performance (e.g., higher classification accuracy) on examples with that spurious attribute. For example, a model that majorly relies on the presence of an emergency vehicle to detect an accident, would have a lower accuracy in detecting accidents when there are no emergency vehicles around.

We use the pipeline depicted in Figure 7.2 to (1) find such spurious attributes for a class of interest if the spurious attributes are unknown, and (2) measure how much each spurious attribute negatively affects the model.

Spurious Detection. For the case where the spurious attribute is unknown, we first use an open-vocabulary detector, OWL-ViT [121], to detect potential spurious attributes for examples in the fine-tuning data. We use the synsets of object names in Visual Genome [95] as our list of attributes to detect after removing objects that are classes of the fine-tuning data.

Spurious Evaluation. We define $\delta(\mathcal{D}, s)$ as the model accuracy discrepancy between examples in dataset $\mathcal{D}$ with the attribute s and those without it.

$$\delta(\mathcal{D}, s) = acc(\mathcal{D}|s = 1) - acc(\mathcal{D}|s = 0). \tag{7.7}$$

Attributes detected in the fine-tuning dataset can then be ranked by their accuracy discrepancy scores. The higher the discrepancy, the more this attribute could harm the generalization performance of the pretrained model. Since model failure modes and in particular spurious correlations are often specific to the class [138], for the ImageNet studies we compute and sort the discrepancy scores per class. While the drop in accuracy with the absence of the spurious attributes are good indicators of spurious correlations, such drops may also happen for healthy attributes that are part of the class definition (e.g., the yellow color for taxi cabs albeit not all taxis are yellow).

Thus, for practical usages of our approach, we imagine this step to involve some miminal human investigation from domain experts or ML practitioners to judge whether the attribute is healthy or a potential spurious correlation. Humans can make this call based on their domain knowledge or one of the vision interpretability techniques (e.g, Grad-CAM, Integrated Gradients etc.). Nevertheless, this kind of supervision is considerably more lightweight than annotating attributes or manually inspecting individual examples. Table 7.3 shows several examples of previously unknown spurious correlations we found for CLIP in ImageNet.

7.4 Experiments

7.4.1 Backbones

CLIP uses two main groups of visual backbones, ResNets (RN) and Visual Transformers (ViT), and reported model performance separately for models with these two types of backbones in [150]. In particular, ResNet-50 (RN50) and ViT-L/14@336px [1] are used as the prototypes of

[1] ViT-L/14@336px refers to ViT-L/14 model fine-tuned on 336-by-336 pixel input images.

Table 7.1: Statistics of the Waterbirds training data.

	Land	Water
Landbirds	3498	184
Waterbirds	56	1057

these two groups of models. Therefore, we follow [150] and study CLIP models with RN50 and ViT-L/14@336px visual backbones in our experiments.

For all experiments, we freeze both the language and vision encoders and only fine-tune the projection layers. Keeping both encoders intact is not only more lightweight but also resulted in better overall and worst-group accuracy for all studied datasets in our preliminary experiments.

7.4.2 Datasets

Waterbirds. Waterbirds [155] is the most commonly used benchmark dataset for studying spurious correlations. It combines birds segmented from the CUB dataset [195] and the background in dataset [234] in an imbalanced way such that the background can be used as a spurious attribute for bird classification. Table 7.1 shows the sample size of each class-background combination in the Waterbirds training set. As landbirds appear more with land background and waterbirds are more often on water background in the training set, models fine-tuned on this dataset often learn to rely on the background instead of the birds.

ImageNet-1K. (author?) found that some features are spuriously correlated with some categories in ImageNet-1K [153]. For example, 55% of training examples in the “Rhodesian ridgeback" class can be correctly classified by a robust ResNet-50 model but the accuracy drops significantly to 24% when the dogs are not wearing a collar. We use the spurious detection pipeline shown in Figure 7.2 to find top-5 attributes with the highest accuracy discrepancy on CLIP for each class and attribute, and then rank the top attributes from all classes. Based on our inspection of the top attributes, we find a number of previously unknown spurious attributes learned by CLIP-RN50 with ImageNet as shown in Table 7.3.

Out of this list, in the mitigation experiments we choose to mitigate the first major spurious correlation that has a high accuracy discrepancy: Baby pacifier class where the spurious attribute is baby face. CLIP accuracy drops by 69.1% for classifying baby pacifiers when there is no baby in the image. Note that since the validation set for ImagenNet contains only 50 images per class, we run the spurious correlation detection and evaluation stages on the training data instead, while mitigation results are presented for the test data. Figures 7.3 and 7.6 show further evidence of the pre-trained model focusing on the spurious attributes rather than the class itself.

Another dataset we considered for evaluation is CelebA [109] for the task of hair color classification. Previous work [117, 155] has shown that models trained on such data can have a lower accuracy for small groups defined by the gender attribute such as men with blond hair, since this group has a low representation in the training data. It turns out however that model accuracy does not degrade for this group using CLIP model, which is why we do not present results on CelebA in this paper.

7.4.3 Metrics

We use the following metrics to evaluate the **predictions** and **explanations** of each model. We argue that only by obtaining high performance in both aspects, an algorithm can be proven to address the spurious correlations and that the correct model predictions are "*right for the right reasons*".

1. **Average Accuracy.** Classification accuracy averaged over classes on the test set. For the Waterbirds dataset, the test data is enriched and balanced to improve the accuracy of the evaluation, but this can lead to a discrepancy between the distribution of the test data and the training data. Following previous works, we report the adjusted average accuracy suggested by [155], which weights the test accuracy of each group by their sizes in the training data.

2. **Worst-group Accuracy.** The lowest model accuracy across groups as defined by the

spurious attribute and the class of interest.

3. **Adjusted Intersection-over-Union (AIoU).** Previous works have used binary attribute maps to compute an Intersection-over-Union (IoU) score with the ground-truth bounding box [137]. While IoU is a standard metric for object localization, using the standard IoU to evaluate the quality of attribute maps can be less reliable because the score highly depends on the threshold used for binarizing the attribute maps. To circumvent threshold dependency, we adapt the formulation such that it instead uses a min operator (instead of the binary intersection) between a bounding box B_y and an explanation map M_y, where y is the ground truth class. Similarly, we use a max operator (instead of binary union) in the denominator between the bounding box and the map.

$$\begin{gathered}\mathrm{IoU}(M, B) = \frac{\sum_{j,k} \min(M_{jk}, B_{jk})}{\sum_{j,k} \max(M_{jk}, B_{jk})}, \\ 0 \leq j \leq h, 0 \leq k \leq w.\end{gathered} \tag{7.8}$$

Equation 7.8 measures the alignment between an explanation map and the ground truth bounding box but it does not take into consideration that despite a good alignment with the bounding box for true class, the explanation maps of other classes may still span across the bounding box of the ground truth class. Therefore, we use a definition of IoU that adjusts its denominator to include the class whose explanation map most intersects with the ground truth bounding box.

$$\mathrm{AIoU} = \frac{\mathrm{IoU}(M_y, B_y)}{\mathrm{IoU}(M_y, B_y) + \max\limits_{y' \in [C \setminus y]} \mathrm{IoU}(M_{y'}, B_y)}. \tag{7.9}$$

In our experiments, we used GradCAM [162] for the explanation maps. While GradCAM explanations may not be perfectly aligned with the model's attention, their usage has shown practical benefits for model debugging [222, 168, 117].

Table 7.2: Accuracy of different groups of Waterbirds on pre-trained ResNet- and Transformer-based CLIP models.

(RN50)	Land	Water
Landbirds	93.44%	44.92%
Waterbirds	59.03%	91.59%

(ViT-L/14@336px)	Land	Water
Landbirds	99.29%	90.20%
Waterbirds	33.96%	55.61%

7.4.4 Baselines

We compare our approach with pre-trained CLIP [150], fine-tuned CLIP with the training dataset in hand using the original contrastive vision and language loss as described in Equation 7.1, Empirical Risk Mimimization (ERM), and Group DRO [155]. Group DRO is a distributionally robust optimization approach that minimizes worst-group loss and uses strong regularization. The methods requires attribute annotations to define groups being used during optimization. ERM instead is the standard empirical risk minimization technique for minimizing classification loss.

Reproducibility. Both CLIP models (CLIP-RN50 and CLIP-ViT) and prompt templates we use in our experiments are officially released by OpenAI[2][150]. The Waterbirds dataset is from the WILDS library [92]. We used the SGD optimizer for all the experiemnts, and tuned the learning rates and weight decays for ERM, GroupDRO and CLIP-based loss (CLIP-finetuning and our method) separately. Our method uses learning rate 1e-5 with weight decay 1e-4. The code will be publicly available upon publication.

[2] https://github.com/openai/CLIP

Table 7.3: Spurious correlations found for CLIP RN50 on ImageNet.

Class	Spurious Attribute	Confused Class	Acc. Discrepancy
baby pacifier	baby	water bottle	62.1%
can opener	can	letter opener	45.2%
eraser	hand	pencil case	18.5%
whistle	ring	padlock	15.2%
pencil sharpener	pencil	pencil case	8.37%

Table 7.4: Results of fine-tuning CLIP with Waterbirds. Average and worst-group performance is evaluated on the test set with models early stopped at the *highest worst-group accuracy* on the validation set. Worst groups: Landbird on water for RN50; Waterbird on land for ViT.

Model	ResNet-50				ViT-L/14@336px			
	Accuracy		AIoU		Accuracy		AIoU	
	Avg.	Worst-group	Avg.	Worst-group	Avg.	Worst-group	Avg.	Worst-group
Pre-trained CLIP	**90.8%**	44.9%	0.507	0.479	88.5%	34.0%	0.579	0.551
Fine-tuned CLIP	81.3%	**77.1%**	0.510	0.128	**97.2%**	89.7%	0.687	0.697
ERM	**93.5%**	54.4%	0.514	0.139	96.8%	58.1%	0.636	0.680
Group DRO	83.3%	73.7%	0.509	0.274	94.1%	**90.8%**	0.669	0.644
Ours($\mathcal{L}_{lc}$+$\mathcal{L}_{vc}$+$\mathcal{L}_{ls}$)	84.7%	**77.5%**	**0.628**	**0.499**	**97.1%**	89.7%	**0.698**	**0.711**
Ours($\mathcal{L}_{lc}$+$\mathcal{L}_{vc}$+$\mathcal{L}_{vs}$)	83.2%	**77.5%**	**0.654**	**0.587**	96.9%	**90.5%**	**0.716**	**0.709**

7.4.5 Spurious Correlation Detection Results

Waterbirds. Table 7.2 shows model accuracy across the four groups as defined by class and spurious attribute definitions. The underlined groups show the worst-group accuracies for each model. For both models, there is a high accuracy discrepancy between groups from the same class. Figures 7.4 and 7.5 show examples of explanations from Pre-trained CLIP where explanations do not overlap with birds.

Imagenet. Table 7.3 shows examples of prominent spurious correlations found for Pre-trained CLIP RN50. It is interesting to see how the found spurious attributes are concepts that are indeed highly related to the class but not necessarily part of the class definition. The natural co-occurence of these concepts leads the model to incorrectly rely rather on the attribute as shown in Figure 7.3.

7.4.6 Spurious Correlation Mitigation Results

Waterbirds. Table 7.4 summarizes our results on the Waterbirds dataset for both Resnet-50 and ViT-L/14@336px. Our method of mitigating spurious correlations through language has the best worst-group accuracy for ResNet-50 and second-best worst-group accuracy for ViT, maintaining a competitive average accuracy. What is of most interest from a mitigation perspective, is that the model ability to be right for the right reasons is indeed better for our method as indicated by the AIoU scores. These results are also qualitatively confirmed by visual explanation maps as shown in Figures 7.4 and 7.5, demonstrating that (i) the spurious correlation is present on the first place (pre-trained CLIP), (ii) it persists in the explanation maps of GroupDRO despite this method being competitive in both worst-group and average accuracy, and (iii) it is visibly alleviated though our approach whose explanations align with the available ground truth segmentations for the dataset. When comparing the two different variants of our method using the spurious language loss and image loss, we observe that the spurious image loss leads to better AIoU scores potentially because decorrelation is easier in the image representation, albeit for this method to work reliable attribute annotations are required. Using the spurious language loss is however still appealing with respect to both worst-group accuracy and AIoU. Note that implicitly, this method, and generally mitigating spurious correlations through language, relies on the capability of the model to map the spurious attribute from language to vision, which may not always be the case for the pre-trained vision-language models. The spurious attributes studied in this paper are based on the language concepts that are perhaps well-learned and mapped in a multi-modal way in CLIP (e.g., baby, water, land) but in other cases of less-frequent or domain-specific attributes, using the spurious image loss may be a more realistic avenue.

When comparing these findings between the two different model backbones we observe that AIoU scores for the ViT model are higher for all methods than their corresponding ResNet versions, indicating that the larger transformer-based model is perhaps more prone to improve upon using such mitigation techniques or even standard fine-tuning.

ImageNet-1K. Here, we choose one of the most spurious correlations we found for the

Table 7.5: Results of fine-tuning CLIP-RN50 with a subset of ImageNet classes, “baby pacifier" and “water bottle". Both average and worst-group performance are evaluated with models early stopped at the *highest worst-group accuracy* on the validation set.

Class 680	Accuracy		AIoU	
Baby Pacifier	Avg.	Worst	Avg.	Worst
Pre-trained	73.7%	30.8%	0.651	0.380
Fine-tuned	94.1%	91.7%	0.650	0.571
ERM	**94.9%**	**96.2%**	0.661	0.454
Group DRO	89.6%	93.1%	0.661	0.568
Ours($\mathcal{L}_{lc}$+$\mathcal{L}_{vc}$+$\mathcal{L}_{ls}$)	**94.9%**	**96.2%**	**0.720**	**0.645**

CLIP ResNet50: the baby pacifier class where the spurious attribute is baby face. The accuracy discrepancy between cases when there is a baby and no baby in the image is 69.2% in the validation set, with a worst-group accuracy of 30.8%. The most confusing class for baby pacifier is water bottle. For all methods, we then fine-tune the CLIP RN50 model with the training data from these two classes: baby pacifier and water bottle to understand if such an isolated mitigation could positively align the model. In Table 7.5 we see that in terms of both average accuracy and worst-group accuracy, the baseline ERM method performs just as well as our methods. However, since the test dataset in this case is rather small (only 50 images per class), it is useful to also look at the alignment of explanations. Figure 7.6 illustrates this visually, highlighting that GradCAM maps are not focused on the baby face for our approach, which is the case for other methods. The same result is confirmed by the higher AIoU scores.

Limitations. While the method proposed here shows promising results for mitigating spurious correlations, learning pipelines often face a combination of problems that go beyond spurious features and involve other out-of-distribution shifts. We illustrate these concerns through a running example from ImageNet in Appendix 9.6.2 and show that current decorrelation methods may not be sufficient when models deal with issues such as high concept variation, insufficient data, label noise, or visual commonalities between spurious and non-spurious features.

7.5 Conclusion and Future Work

We proposed a language-based approach to mitigate spurious correlations of CLIP, as a contrastive learning vision and language model. Our focus on mitigations that can be initiated through language is motivated by the fact that spurious attribute annotations may not always be available. The contrastive loss function formulation guiding the spurious attribute decorrelation is applied at fine-tuning time and is effective even when the language and image encoders are excluded from the fine-tuning process. Besides the computational convenience, this is a promising finding speaking to the foundational nature of the larger representations. The work opens up several questions for future research, including the scalability of such methods when mitigating several spurious correlations at the same time. While this work focused on spurious correlations for classification tasks, studying the problem from a representational bias perspective and how spurious correlations may feed issues in representation fairness is an important relevant direction with several societal implications. Finally, we see opportunities in further leveraging model multi-modality and language-initiated mitigation actions to either generate teaching samples for mitigations or high-level instructions for the model to follow.

Acknowledgment. This research was partially supported by Cisco Systems and the National Science Foundation CAREER Award 2146492.

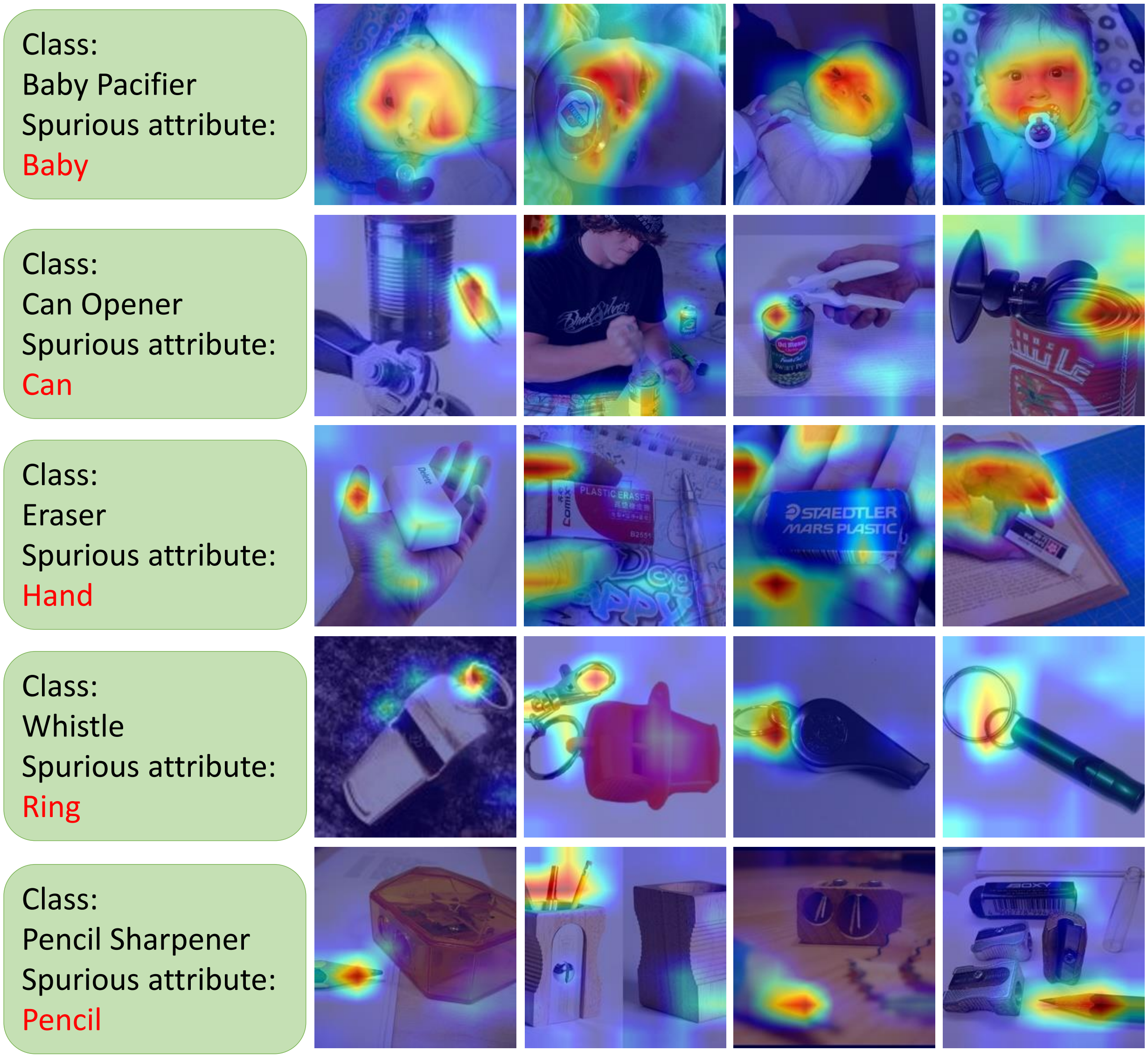


Figure 7.3: GradCAM explanations for cases when Pre-trained CLIP RN50 relies on the spurious classification described in Table 7.3.

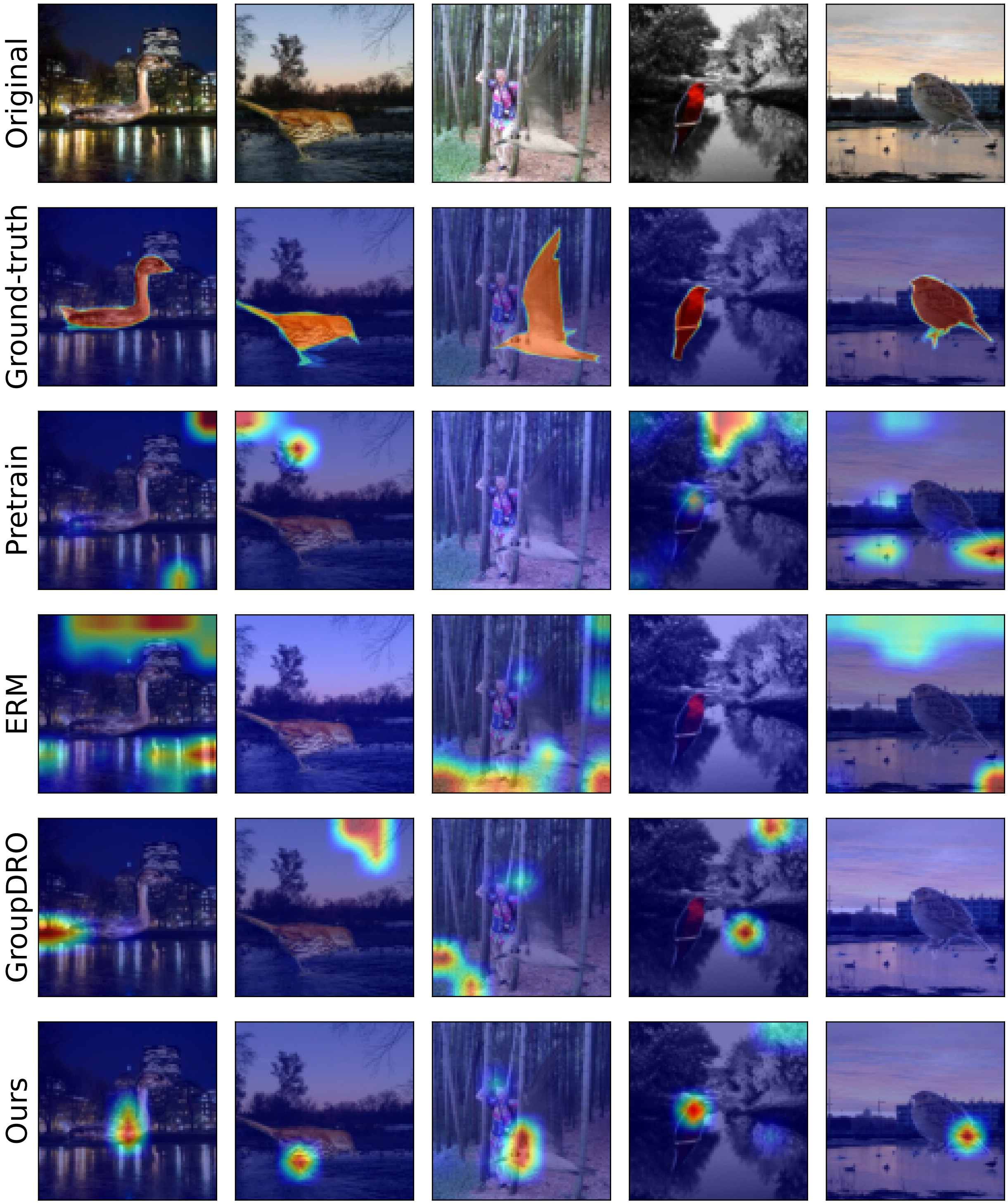


Figure 7.4: GradCAM explanations for different approaches based on CLIP RN50 for the Waterbirds dataset.

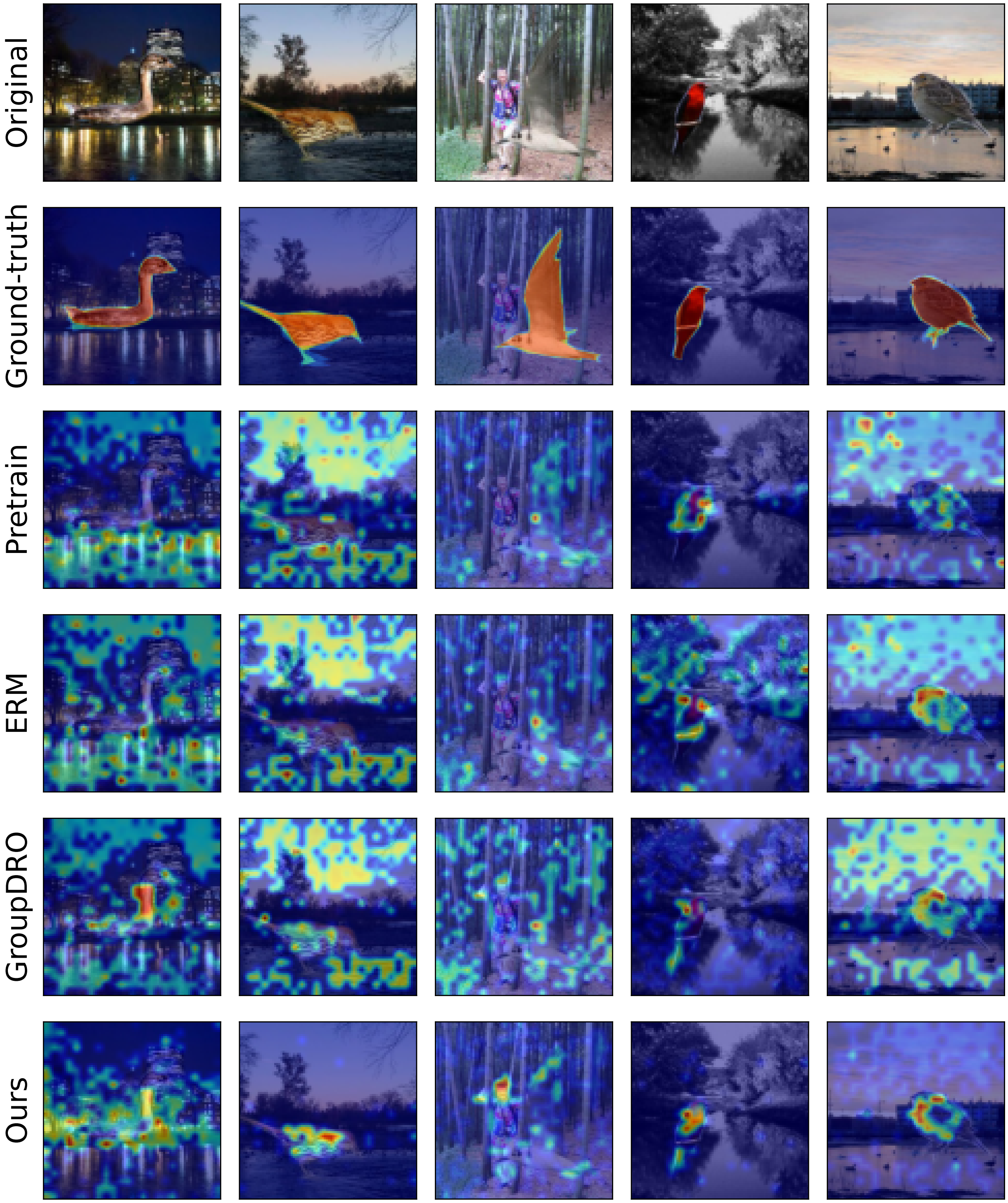


Figure 7.5: GradCAM explanations for different approaches based on CLIP ViT-L/14@336px for the Waterbirds dataset.

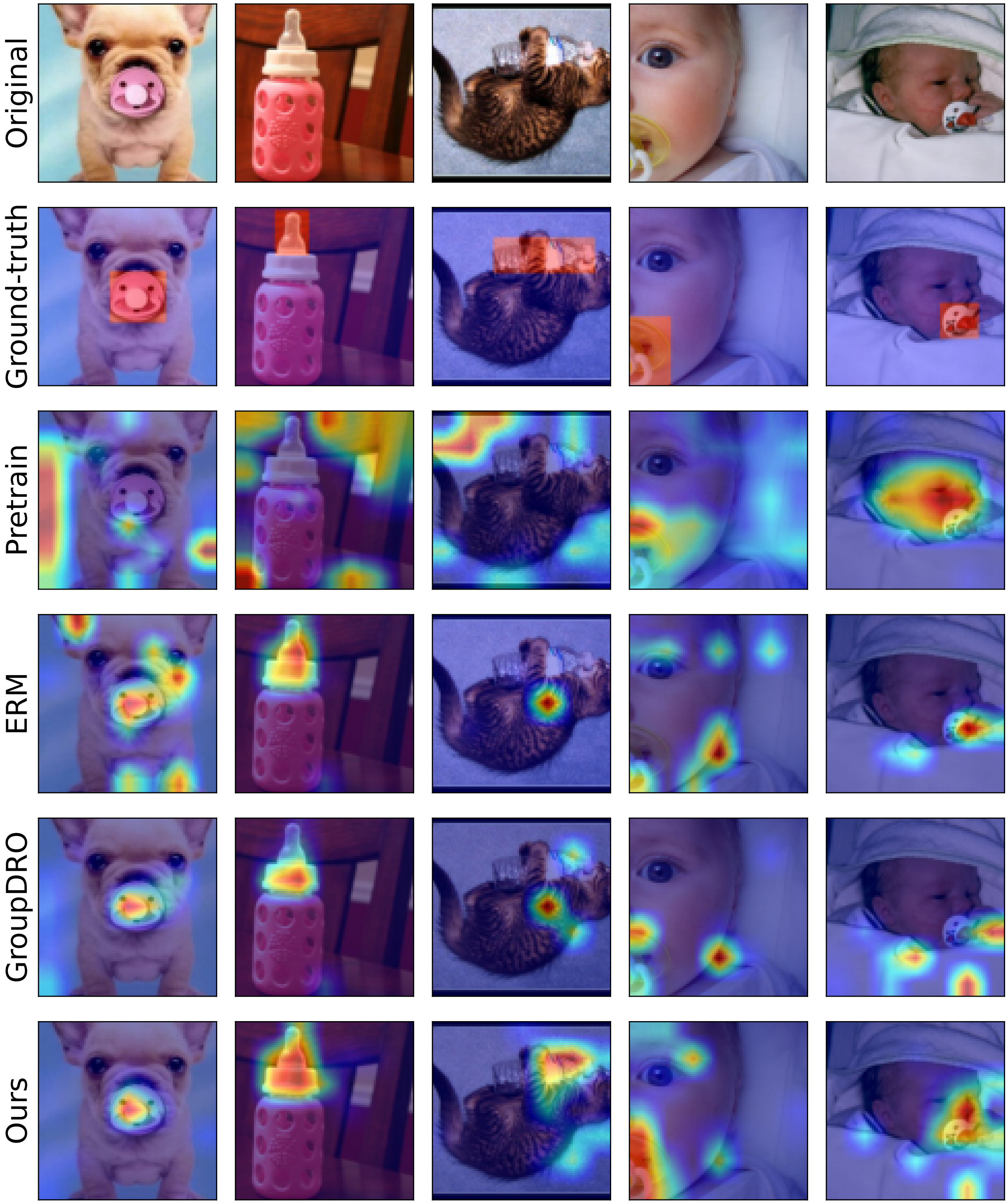


Figure 7.6: GradCAM explanations for different approaches based on CLIP RN50 for the ImageNet dataset.

CHAPTER 8

Conclusion and Future Work

In conclusion, my research efforts have yielded significant advancements in the field of improving the efficiency, robustness, and generalization performance of deep learning models.

Chapter 2 introduced CREST, a scalable framework that provides rigorous theoretical guarantees for identifying valuable training examples in non-convex models, specifically deep networks. By modeling the non-convex loss as a piece-wise quadratic function and extracting mini-batch coresets for each sub-region, CREST helps deep learning models achieve faster convergence. I demonstrated the effectiveness of our approach through extensive experiments on vision and NLP tasks.

Chapter 3 introduced SmallToLarge (S2L), a data selection method for efficiently fine-tuning large language models (LLMs). S2L leverages training trajectories from smaller proxy models to select representative subsets, drastically reducing data requirements while maintaining performance. Experiments on mathematical reasoning and clinical text summarization demonstrated its scalability and effectiveness, with significant improvements in efficiency for specialized domains.

In Chapter 4, I introduced EPIC, an efficient defense mechanism against various data poisoning attacks. My analysis revealed that only a small number of poisons with gradients close to the target's can cause successful attacks. By dropping the isolated examples in the gradient space and training on large gradient clusters of each class, EPIC significantly reduces the success rate of state-of-the-art targeted attacks. Importantly, my proposed algorithm EPIC is effective against strong poisoning attacks and seamlessly integrates into standard deep learning pipelines.

In Chapter 5, I investigated the learning of spurious features in neural networks during gradient-based training. My analysis showed that large groups of examples with spurious features are separable based on the model's output early in training. Moreover, when spurious features have a low noise-to-signal ratio, the network's output becomes predominantly determined by these features, rendering it invariant to core features. To address this, I proposed the SPARE algorithm, which effectively separates and balances these groups through clustering and importance sampling. Experimental results demonstrated superior performance in terms of worst-group accuracy on various datasets, including its applicability to discovering and mitigating spurious correlations in Restricted ImageNet.

In Chapter 6, I introduced PDE (Progressive Data Expansion), a method designed to improve robustness against spurious correlations while maintaining data efficiency. PDE employs a two-stage pipeline: it starts with a balanced subset of examples to provide an unbiased foundation for training, then progressively expands the training set to include more diverse examples. This strategy helps guide models toward learning core features over spurious ones. Experimental results demonstrated state-of-the-art worst-group accuracy on datasets like CivilComments-WILDS and CelebA, showcasing PDE's ability to address biases while preserving training efficiency.

Chapter 7 focused on mitigating spurious correlations in multimodal models like CLIP. This work proposed a contrastive learning approach that uses language to identify and decorrelate spurious attributes during fine-tuning. Evaluations on datasets like Waterbirds and ImageNet showed substantial improvements in worst-group accuracy and highlighted the method's ability to realign model attention toward core features, improving both robustness and interpretability.

Building on my contributions, several promising research directions can further advance the field of data efficiency and robustness. First, expanding data selection frameworks to handle large, heterogeneous, and multimodal datasets presents an exciting opportunity. By integrating diverse data types such as text, images, and structured data, these methods can optimize efficiency while maintaining domain relevance in increasingly complex tasks. Second, mitigating spurious correlations in generative AI and multimodal systems offers a critical avenue for improving fairness and reliability. Extending the techniques developed in my CLIP spurious correlation work, future research could address challenges in tasks such as text-to-image generation and cross-modal reasoning. Third, developing dynamic, real-time data selection methods that adaptively update subsets as new data becomes available is essential for applications requiring continuous learning, such as online platforms and autonomous systems.

CHAPTER 9

Appendices

9.1 Appendix for Chapter 2

9.1.1 Proofs

We assume that the stochastic gradients are unbiased and have a bounded variance, i.e.,

$$\mathbb{E}_{i\in V}[\|\mathbf{g}_{t,i}-\mathbf{g}_{t,V}\|]=\mathbb{E}[\|\boldsymbol{\zeta}_t\|]=0, \qquad \mathbb{E}_{i\in V}[\|\mathbf{g}_{t,i}-\mathbf{g}_{t,V}\|^2]=\mathbb{E}[\|\boldsymbol{\zeta}_t\|^2]\leq\sigma^2. \tag{9.1}$$

Also assume that the function $\mathcal{L}$ is L-gradient Lipschitz, i.e.,

$$\|\nabla\mathcal{L}(\boldsymbol{w}_1)-\nabla\mathcal{L}(\boldsymbol{w}_2)\|\leq L\|\boldsymbol{w}_1-\boldsymbol{w}_2\|, \quad \forall\boldsymbol{w}_1,\boldsymbol{w}_2\in\mathcal{W}. \tag{9.2}$$

Then, we have that:

$$|\mathcal{L}(\boldsymbol{w}_1)-\mathcal{L}(\boldsymbol{w}_2)-\langle\nabla\mathcal{L}(\boldsymbol{w}_2),\boldsymbol{w}_1-\boldsymbol{w}_2\rangle|\leq\frac{L}{2}\|\boldsymbol{w}_1-\boldsymbol{w}_2\|^2, \quad \forall\boldsymbol{w}_1,\boldsymbol{w}_2\in\mathcal{W}. \tag{9.3}$$

We can write the gradient descent updates when training on mini-batch coresets found by Spare, as follows:

$$\boldsymbol{w}_{t+1}\leftarrow\boldsymbol{w}_t-\eta_t(\nabla\mathcal{L}(\boldsymbol{w}_t)+\tilde{\boldsymbol{\zeta}}_t), \qquad s.t. \qquad \tilde{\boldsymbol{\zeta}}_t=\boldsymbol{\zeta}_t+\boldsymbol{\xi}_t \tag{9.4}$$

where $\boldsymbol{\zeta}_t$ is the error of random subset V_p in capturing the full gradient, and $\boldsymbol{\xi}_t$ is the error of mini-batch coreset S_l^p in capturing the gradient of V_p.

We build on the analysis of [61] and characterize the effect of the coreset gradient error on the convergence. From Eq. (9.3), (9.4) we have:

$$\mathcal{L}(\boldsymbol{w}_{t+1}) \tag{9.5}$$

$$\leq \mathcal{L}(\boldsymbol{w}_t) + \langle \nabla\mathcal{L}(\boldsymbol{w}_t), \boldsymbol{w}_{t+1} - \boldsymbol{w}_t \rangle + \frac{L}{2}\eta_t^2 \|\nabla\mathcal{L}(\boldsymbol{w}_t) + \tilde{\boldsymbol{\zeta}}_t\|^2 \tag{9.6}$$

$$\leq \mathcal{L}(\boldsymbol{w}_t) - \eta_t \left\langle \nabla\mathcal{L}(\boldsymbol{w}_t), \nabla\mathcal{L}(\boldsymbol{w}_t) + \tilde{\boldsymbol{\zeta}}_t \right\rangle + \frac{L}{2}\eta_t^2 \|\nabla\mathcal{L}(\boldsymbol{w}_t) + \tilde{\boldsymbol{\zeta}}_t\|^2 \tag{9.7}$$

$$= \mathcal{L}(\boldsymbol{w}_t) - \eta_t \|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2 - \eta_t \left\langle \nabla\mathcal{L}(\boldsymbol{w}_t), \tilde{\boldsymbol{\zeta}}_t \right\rangle + \frac{L}{2}\eta_t^2 \left[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2 + 2\left\langle \nabla\mathcal{L}(\boldsymbol{w}_t), \tilde{\boldsymbol{\zeta}}_t \right\rangle + \|\tilde{\boldsymbol{\zeta}}_t\|^2 \right] \tag{9.8}$$

$$= \mathcal{L}(\boldsymbol{w}_t) - (\eta_t - \frac{L}{2}\eta_t^2) \|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2 - (\eta_t - L\eta_t^2) \left\langle \nabla\mathcal{L}(\boldsymbol{w}_t), \tilde{\boldsymbol{\zeta}}_t \right\rangle + \frac{L}{2}\eta_t^2 \|\tilde{\boldsymbol{\zeta}}_t\|^2 \tag{9.9}$$

For $\eta_t < 2/L$, we have $\eta_t - L\eta_t^2/2 > 0$. Summing up the above inequalities and re-arranging the terms, we obtain:

$$\sum_{t=1}^{N} (\eta_t - \frac{L}{2}\eta_t^2) \|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2 \tag{9.10}$$

$$\leq \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}(\boldsymbol{w}_{N+1}) - \sum_{t=1}^{N} (\eta_t - L\eta_t^2) \left\langle \nabla\mathcal{L}(\boldsymbol{w}_t), \tilde{\boldsymbol{\zeta}}_t \right\rangle + \frac{L}{2} \sum_{t=1}^{N} \eta_t^2 \|\tilde{\boldsymbol{\zeta}}_t\|^2 \tag{9.11}$$

$$\leq \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - \sum_{t=1}^{N} (\eta_t - L\eta_t^2) \left\langle \nabla\mathcal{L}(\boldsymbol{w}_t), \tilde{\boldsymbol{\zeta}}_t \right\rangle + \frac{L}{2} \sum_{t=1}^{N} \eta_t^2 \|\tilde{\boldsymbol{\zeta}}_t\|^2, \tag{9.12}$$

$$= \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - \sum_{t=1}^{N} (\eta_t - L\eta_t^2) \langle \nabla\mathcal{L}(\boldsymbol{w}_t), \boldsymbol{\zeta}_t + \boldsymbol{\xi}_t \rangle + \frac{L}{2} \sum_{t=1}^{N} \eta_t^2 \|\boldsymbol{\zeta}_t + \boldsymbol{\xi}_t\|^2, \tag{9.13}$$

$$= \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - \sum_{t=1}^{N} (\eta_t - L\eta_t^2) \langle \nabla\mathcal{L}(\boldsymbol{w}_t), \boldsymbol{\zeta}_t + \boldsymbol{\xi}_t \rangle + \frac{L}{2} \sum_{t=1}^{N} \eta_t^2 (\|\boldsymbol{\zeta}_t^2\| + \|\boldsymbol{\xi}_t^2\| + 2\langle \boldsymbol{\zeta}_t, \boldsymbol{\xi}_t \rangle), \tag{9.14}$$

where $\mathcal{L}^*$ is the optimal solution and Eq. (9.12) follows from the fact that $\mathcal{L}(\boldsymbol{w}_{N+1}) \geq \mathcal{L}^*$. Taking expectations (with respect to the history Ψ_N of the generated random process) on both sides of Eq. (9.14) and noting that $\mathbb{E}[\|\boldsymbol{\zeta}_t\|] = 0$, and $\mathbb{E}[\|\boldsymbol{\zeta}_t\|^2] \leq \sigma^2$, and $\mathbb{E}[\langle \nabla\mathcal{L}(\boldsymbol{w}_t), \boldsymbol{\zeta}_t \rangle \,|\Psi_{t-1}] = 0$, and $\mathbb{E}[\langle \boldsymbol{\zeta}_t, \boldsymbol{\xi}_t \rangle \,|\Psi_{t-1}] = 0$ (since $\nabla\mathcal{L}(\boldsymbol{w}_t)$ and $\boldsymbol{\xi}_t$ and $\boldsymbol{\zeta}_t$ are independent), we obtain:

$$\begin{aligned}\sum_{t=1}^{N}(\eta_t - \frac{L}{2}\eta_t^2)\mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2] \leq \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - &\sum_{t=1}^{N}(\eta_t - L\eta_t^2)\mathbb{E}_{\Psi_N}[\langle\nabla\mathcal{L}(\boldsymbol{w}_t), \boldsymbol{\xi}_t\rangle] \\ &+ \sum_{t=1}^{N}\frac{L}{2}\eta_t^2(\frac{\sigma^2}{r} + \mathbb{E}_{\Psi_N}[\|\boldsymbol{\xi}_t\|^2]).\end{aligned} \tag{9.15}$$

Next, we analyze convergence under two cases: (1) where $\mathbb{E}[\|\boldsymbol{\xi}_t\|] \leq \epsilon\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|$, and (2) where $\mathbb{E}[\|\boldsymbol{\xi}_t\|] \leq \epsilon$.

Case 1. Assuming $\mathbb{E}[\|\boldsymbol{\xi}_t\|] \leq \epsilon\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|$ for $0 \leq \epsilon < 1$. With $1/L \leq \eta_t < 2/L$, we have $\eta_t - L\eta_t^2 \leq 0$. Hence,

$$\begin{aligned}\sum_{t=1}^{N}(\eta_t - \frac{L}{2}\eta_t^2)\mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2] \leq \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - &\sum_{t=1}^{N}(\eta_t - L\eta_t^2)\epsilon\mathbb{E}_{\Psi}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2] \\ &+ \sum_{t=1}^{N}\frac{L}{2}\eta_t^2(\frac{\sigma^2}{r} + \epsilon^2\mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2]).\end{aligned} \tag{9.16}$$

Hence,

$$\sum_{t=1}^{N}\left(\eta_t - \frac{L}{2}\eta_t^2 + \epsilon(\eta_t - L\eta_t^2) - \frac{L}{2}\eta_t^2\epsilon^2\right)\mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2] \leq \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* + \frac{L\sigma^2}{2r}\sum_{t=1}^{N}\eta_t^2, \tag{9.17}$$

$$\sum_{t=1}^{N}\left((1+\epsilon)\eta_t - \frac{L}{2}(1+\epsilon)^2\eta_t^2\right)\mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2] \leq \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* + \frac{L\sigma^2}{2r}\sum_{t=1}^{N}\eta_t^2, \tag{9.18}$$

and we get:

$$\mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2] \leq \frac{1}{\sum_{t=1}^{N}(1+\epsilon)\eta_t - \frac{L}{2}(1+\epsilon)^2\eta_t^2}\left[\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* + \frac{\sigma^2 L}{2r}\sum_{t=1}^{N}\eta_t^2\right] \tag{9.19}$$

If $\eta_t < (1+\epsilon)/(\frac{L}{2}(1+\epsilon)^2) = 2/L(1+\epsilon)$, then $(1+\epsilon) - \frac{L}{2}(1+\epsilon)^2\eta_t > 0$ and we have:

$$\mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2] \leq \frac{1}{\sum_{t=1}^{N}\eta_t}\left[2(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*) + \frac{\sigma^2 L}{r}\sum_{t=1}^{N}\eta_t^2\right]. \tag{9.20}$$

For a random iterate R of a run of the algorithm that is selected with probability $(2(1+\epsilon)\eta_t - L(1+\epsilon)^2\eta_t^2)/\sum_{t=1}^{N}(2(1+\epsilon)\eta_t - L(1+\epsilon)^2\eta_t^2)$, we have that

$$\begin{aligned}\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{w}_R)\|^2] &= \mathbb{E}_{R,\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_R)\|^2] \\ &= \frac{\sum_{t=1}^{N}(2(1+\epsilon)\eta_t - L(1+\epsilon)^2\eta_t^2)\mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2]}{\sum_{t=1}^{N}(2(1+\epsilon)\eta_t - L(1+\epsilon)^2\eta_t^2)} \\ &= \mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2]\end{aligned} \tag{9.21}$$

Hence, for $\eta_t = \eta$ we get:

$$\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{w}_R)\|^2] \leq \frac{1}{N\eta}\left[2(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*) + \frac{\sigma^2 L}{r}N\eta^2\right]. \tag{9.22}$$

For $\eta = \min\{\frac{1}{L}, \frac{\tilde{D}\sqrt{r}}{\sigma\sqrt{N}}\}$, and $\tilde{D} > 0$, we get

$$\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{w}_R)\|^2] \leq \frac{1}{N\eta}\left[2(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)\right] + \frac{\sigma^2 L}{r}\eta \tag{9.23}$$

$$\leq \frac{2(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{N}\max\{L, \frac{\sigma\sqrt{N}}{\tilde{D}\sqrt{r}}\} + \frac{\sigma^2 L}{r}\frac{\tilde{D}\sqrt{r}}{\sigma\sqrt{N}} \tag{9.24}$$

$$\leq \frac{2L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{N} + \left(L\tilde{D} + \frac{2(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{\tilde{D}}\right)\frac{\sigma}{\sqrt{rN}} \tag{9.25}$$

Replacing the optimal value $\tilde{D} = \sqrt{2(\mathcal{L}(\boldsymbol{w}_1) - \mathcal{L}^*)/L}$, we get

$$\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{w}_R)\|^2] \leq \mathcal{B}_N := \frac{2L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{N} + \frac{2\sigma\sqrt{2L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}}{\sqrt{rN}} \tag{9.26}$$

Hence, training with SPARE exhibits an $\mathcal{O}(1/\sqrt{rN})$ rate of convergence, compared to $\mathcal{O}(1/\sqrt{mN})$ for mini-batch SGD with mini-batch size $m < r$.

To derive large-deviation properties for a single run of this method, we are interested in the number of iterations required to find a point satisfying $\mathbb{P}[\|\nabla\mathcal{L}(\boldsymbol{w}_R)\|^2 \leq \nu^2] \geq 1 - \frac{1}{\lambda}$. We use Markov's inequality to calculate the probability $\mathbb{P}[\|\nabla\mathcal{L}(\boldsymbol{w}_R)\|^2 \geq \lambda\mathcal{B}_N] \leq \frac{1}{\lambda}$. We get that with probability at least $1-\lambda$, at least one iteration of a single run of the algorithm visits a ν-stationary point in the following number of iterations:

$$\tilde{\mathcal{O}}\left(\frac{L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{\nu^2}(1 + \frac{\sigma^2}{r\nu^2})\right). \tag{9.27}$$

As long as $r \leq \sigma^2/\nu^2$ increasing the size of the random subsets r used by Spare will reduce the number of iterations linearly, while not increasing the total number of stochastic gradient queries.

Case 2. Assuming $\mathbb{E}[\|\boldsymbol{\xi}_t\|] \leq \epsilon < \nu^2$. For $1/L \leq \eta_t < 2/L$ we have $\eta_t - L\eta_t^2 < 0$. Hence:

$$\sum_{t=1}^{N}(\eta_t - \frac{L}{2}\eta_t^2)\mathbb{E}_{\Psi}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2] \tag{9.28}$$

$$\leq \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - \sum_{t=1}^{N}(\eta_t - L\eta_t^2)\mathbb{E}_{\Psi}[\langle\nabla\mathcal{L}(\boldsymbol{w}_t), \xi_t\rangle] + \sum_{t=1}^{N}\frac{L}{2}\eta_t^2(\frac{\sigma^2}{r} + \epsilon^2), \tag{9.29}$$

$$\leq \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - \sum_{t=1}^{N}(\eta_t - L\eta_t^2)\mathbb{E}_{\Psi}[|\langle\nabla\mathcal{L}(\boldsymbol{w}_t), \xi_t\rangle|] + \sum_{t=1}^{N}\frac{L}{2}\eta_t^2(\frac{\sigma^2}{r} + \epsilon^2), \tag{9.30}$$

$$\leq \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - \sum_{t=1}^{N}(\eta_t - L\eta_t^2)\epsilon\mathbb{E}_{\Psi}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|] + \sum_{t=1}^{N}\frac{L}{2}\eta_t^2(\frac{\sigma^2}{r} + \epsilon^2) \tag{9.31}$$

$$\leq \mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - \sum_{t=1}^{N}(\eta_t - L\eta_t^2)\epsilon\nabla_{\max} + \sum_{t=1}^{N}\frac{L}{2}\eta_t^2(\frac{\sigma^2}{r} + \epsilon^2), \tag{9.32}$$

where $\nabla_{\max} = \max\{0, \max_{i\in V, \boldsymbol{w}_t\in\boldsymbol{W}} \mathbf{g}_{t,i}\}$. For a random iterate R of a run of the algorithm that is selected with probability $(2\eta_t - L\eta_t^2)/\sum_{t=1}^{N}(2\eta_t - L\eta_t^2)$, we have that

$$\begin{aligned}\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{w}_R)\|^2] &= \mathbb{E}_{R,\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_R)\|^2] \\ &= \frac{\sum_{t=1}^{N}(2\eta_t - L\eta_t^2)\mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2]}{\sum_{t=1}^{N}(2\eta_t - L\eta_t^2)} \\ &= \mathbb{E}_{\Psi_N}[\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|^2]\end{aligned} \tag{9.33}$$

If $\eta = \eta_t$ and $\eta \leq 1/L + 1/2L\nabla_{\max}$, we have $-2(1 - L\eta) \leq 1/\nabla_{\max}$ and we get

$$\mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{w}_R)\|^2] \leq \frac{1}{N\eta(1 - \frac{L}{2}\eta)} \left[\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - N(\eta - L\eta^2)\epsilon\nabla_{\max} + (\frac{\sigma^2}{r} + \epsilon^2)\frac{L}{2}N\eta^2\right] \tag{9.34}$$

$$\leq \frac{2}{N\eta} \left[\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^* - N(\eta - L\eta^2)\epsilon\nabla_{\max} + (\frac{\sigma^2}{r} + \epsilon^2)\frac{L}{2}N\eta^2\right] \tag{9.35}$$

$$= \frac{2}{N\eta} [\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*] - 2(1 - L\eta)\epsilon\nabla_{\max} + (\frac{\sigma^2}{r} + \epsilon^2)L\eta \tag{9.36}$$

$$\leq \frac{2}{N\eta} [\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*] + \epsilon + (\frac{\sigma^2}{r} + \epsilon^2)L\eta \tag{9.37}$$

For $\eta = \min\{\frac{1}{L}, \frac{\tilde{D}}{\sqrt{N(\sigma^2/r + \epsilon^2)}}\}$, and $\tilde{D} > 0$, we get

$$\mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{w}_R)\|^2] \leq \frac{1}{N\eta} [2(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)] + (\frac{\sigma^2}{r} + \epsilon^2)L\eta + \epsilon \tag{9.38}$$

$$\leq \frac{2(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{N} \max\{L, \frac{\sqrt{N(\sigma^2/r + \epsilon^2)}}{\tilde{D}}\} + (\frac{\sigma^2}{r} + \epsilon^2)\frac{L\tilde{D}}{\sqrt{N(\sigma^2/r + \epsilon^2)}} + \epsilon \tag{9.39}$$

$$\leq \frac{2L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{N} + \left(L\tilde{D} + \frac{2(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{\tilde{D}}\right)\frac{\sqrt{\sigma^2/r + \epsilon^2)}}{\sqrt{N}} + \epsilon \tag{9.40}$$

For the optimal value of $\tilde{D} = \sqrt{2(\mathcal{L}(\boldsymbol{w}_1) - \mathcal{L}^*)/L}$, we get

$$\mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{w}_R)\|^2] \leq \mathcal{B}_N := \frac{2L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{N} + \frac{2\sqrt{\sigma^2 + r\epsilon^2}\sqrt{2L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}}{\sqrt{rN}} + \epsilon \tag{9.41}$$

Hence, the number of iterations becomes:

$$\tilde{\mathcal{O}}\left(\frac{L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{\nu^2 - \epsilon}(1 + \frac{\sigma^2 + r\epsilon^2}{r(\nu^2 - \epsilon)})\right) \tag{9.42}$$

Hence, more number of iterations is required. Besides, if $\epsilon \geq \nu^2$, convergence is not guaranteed.

Incorporating τ. Assume c_2 is the error of the coreset in capturing the full gradient at the beginning of the neighborhood. From Eq. (2.10) we know $|\mathcal{L}(\boldsymbol{w}_{t_l} + \boldsymbol{\delta}) - \mathcal{F}^l(\boldsymbol{\delta})| = \rho_{t_l}\mathcal{L}(\boldsymbol{w}_{t_l} + \boldsymbol{\delta}_l)$. Using the quadratic approximation in Eq. (2.6), i.e., $\mathcal{F}^l(\boldsymbol{\delta}) = \frac{1}{2}\boldsymbol{\delta}^T\mathbf{H}_{t_l,S_l}\boldsymbol{\delta} + \mathbf{g}_{t_l,S_l}\boldsymbol{\delta} + \mathcal{L}(\boldsymbol{w}_{t_l})$, and noting that $\mathcal{L}$ can also be modeled by a similar quadratic function for small τ, we get:

$$\rho_{t_l}\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l) = |\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l) - \mathcal{F}^l(\boldsymbol{\delta}_l)| = |\frac{1}{2}\boldsymbol{\delta}_l^T(\mathbf{H}_{t_l,V} - \mathbf{H}_{t_l,S_l})\boldsymbol{\delta}_l + (\mathbf{g}_{t_l,V} - \mathbf{g}_{t_l,S_l})\boldsymbol{\delta}_l| \tag{9.43}$$

$$\geq \left|\frac{1}{2}|\boldsymbol{\delta}_l^T(\mathbf{H}_{t_l,V} - \mathbf{H}_{t_l,S_l})\boldsymbol{\delta}_l| - \|\mathbf{g}_{t_l,V} - \mathbf{g}_{t_l,S_l}\| \cdot \|\boldsymbol{\delta}_l\|\right| \tag{9.44}$$

$$\geq \left|\frac{1}{2}|\boldsymbol{\delta}_l^T(\mathbf{H}_{t_l,V} - \mathbf{H}_{t_l,S_l})\boldsymbol{\delta}_l| - c_2 \cdot \|\boldsymbol{\delta}_l\|\right| \tag{9.45}$$

$$\geq \frac{1}{2}|\boldsymbol{\delta}_l^T(\mathbf{H}_{t_l,V} - \mathbf{H}_{t_l,S_l})\boldsymbol{\delta}_l| - c_2 \cdot \|\boldsymbol{\delta}_l\| \tag{9.46}$$

As long as ρ_{t_l} is small, we can assume that the loss can be well modeled by a quadratic using the Hessian diagonal. Using the Hessian diagonal for both $\mathcal{L}$ and $\mathcal{F}^l$, we have

$$|\boldsymbol{\delta}_l^T(\mathbf{H}_{t_l,S_l} - \mathbf{H}_{t_l,V})\boldsymbol{\delta}_l| = \|\boldsymbol{\delta}_l^T(\mathrm{diag}(\mathbf{H}_{t_l,S_l}) - \mathrm{diag}(\mathbf{H}_{t_l,V}))\| \cdot \|\boldsymbol{\delta}_l\|.$$

Hence,

$$\frac{1}{2}|\boldsymbol{\delta}_l^T(\mathbf{H}_{t_l,S_l} - \mathbf{H}_{t_l,V})\boldsymbol{\delta}_l| = \frac{1}{2}\|\boldsymbol{\delta}^T(\mathrm{diag}(\mathbf{H}_{t_l,S_l}) - \mathrm{diag}(\mathbf{H}_{t_l,V}))\| \cdot \|\boldsymbol{\delta}_l\| \leq \rho_{t_l}\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l) + c_2\|\boldsymbol{\delta}_l\|, \tag{9.47}$$

$$\|\boldsymbol{\delta}_l^T(\mathrm{diag}(\mathbf{H}_{t_l,S_l}) - \mathrm{diag}(\mathbf{H}_{t_l,V}))\| \leq 2\rho_{t_l}\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)/\|\boldsymbol{\delta}_l\| + 2c_2. \tag{9.48}$$

On the other hand, we have that $\nabla\mathcal{F}^l(\boldsymbol{\delta}) = \boldsymbol{\delta}^T\mathbf{H}_{t_l,S_l} + \mathbf{g}_{t_l,S_l}$ and $\nabla\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l) = \boldsymbol{\delta}_l^T\mathbf{H}_{t_l,V} + \mathbf{g}_{t_l,V}$. Hence, we have:

$$\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l) - \nabla\mathcal{F}^l(\boldsymbol{\delta}_l)\| = \|\boldsymbol{\delta}_l^T(\mathbf{H}_{t_l,V} - \mathbf{H}_{t_l,S_l}) + (\mathbf{g}_{t_l,V} - \mathbf{g}_{t_l,S_l})\| \tag{9.49}$$

$$\leq \|\boldsymbol{\delta}_l^T(\mathbf{H}_{t_l,V} - \mathbf{H}_{t_l,S_l})\| + \|\mathbf{g}_{t_l,V} - \mathbf{g}_{t_l,S_l}\| \tag{9.50}$$

$$\leq \|\boldsymbol{\delta}_l^T(\mathbf{H}_{t_l,V} - \mathbf{H}_{t_l,S_l})\| + c_2. \tag{9.51}$$

Therefore, using Hessian diagonal and from Eq. (9.48) and (9.51) we get:

$$\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)-\nabla\mathcal{F}^l(\boldsymbol{\delta}_l)\| \leq \|\boldsymbol{\delta}_l^T(\text{diag}(\mathbf{H}_{t_l,V})-\text{diag}(\mathbf{H}_{t_l,S_l}))\|+c_2 \leq 2\rho_{t_l}\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)/\|\boldsymbol{\delta}_l\|+3c_2 \tag{9.52}$$

Eq. (9.52) shows that for a fixed ρ_{t_l} and loss, if the convex approximation $\mathcal{F}^l$ is valid in a larger neighborhood $\boldsymbol{\delta}_l$, then the error of the Hessian diagonal at the beginning of the neighborhood was smaller and hence the gradient error at the end of the neighborhood is smaller.

From Eq. (9.52) we know that $\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)-\nabla\mathcal{F}^l(\boldsymbol{\delta}_l)\| \leq 2\rho_{t_l}\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)/\|\boldsymbol{\delta}_l\|+3c_2$. Let c_1 be the desired upper-bound on the gradient error at $\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l$. Hence, we wish

$$\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)-\nabla\mathcal{F}^l(\boldsymbol{\delta}_l)\| \leq 2\rho_{t_l}\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)/\|\boldsymbol{\delta}_l\|+3c_2 \leq c_1. \tag{9.53}$$

Hence, for $c_2 \leq c_1/3$ we get:

$$\rho_{t_l} \leq \frac{(c_1-3c_2)\|\boldsymbol{\delta}_l\|}{2\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)}, \qquad \tau \leq \min_{t_l}\rho_{t_l}. \tag{9.54}$$

For $\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)-\nabla\mathcal{F}^l(\boldsymbol{\delta}_l)\| \leq c_1\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)\|$, and $c_2 \leq c_1\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)\|/3$, we have:

$$\rho_{t_l} \leq \frac{(c_1\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)\|-3c_2)\|\boldsymbol{\delta}_l\|}{2\mathcal{L}(\boldsymbol{w}_{t_l}+\boldsymbol{\delta}_l)}, \tag{9.55}$$

For $c_1 = 1$ and $c_2 = \epsilon\|\nabla\mathcal{L}(\boldsymbol{w}_t)\|$, we get Case 1 in the Theorem. We see that when gradient norm is smaller, we should have a smaller error c_2 in capturing the random subset gradients.

9.1.2 Experimental details

Tuning Hyperparameters. We tuned the hyperparameters $\tau \in \{0.1, 0.05, 0.01, 0.005, 0.001\}$, $h \in \{1, 2, 4, 8, 10\}$ and used $\tau = 0.05, 0.01, 0.005, 0.05,\ \ h = 1, 10, 1, 4$ on CIFAR-10, CIFAR-100, TinyImagenet, and SNLI, respectively, as listed in Table 9.3. To determine τ, we calculated the average loss approximation error divided by the training loss, i.e. ρ_{t_l} in Eq.

Table 9.1: Experiment setups.

Dataset	Classes	Train	Network	Parameters	Ful acc
CIFAR-10	10	50k	ResNet-20	0.27M	$92.1_{\pm 0.1}$
CIFAR-100	100	50k	ResNet-18	11M	$75.6_{\pm 0.3}$
TinyImageNet	200	100k	ResNet-50	23M	$66.9_{\pm 0.1}$
SNLI	3	570k	RoBERTa	123M	$92.9_{\pm 0.2}$

Table 9.2: Relative error (%) with 20% of the full training budget (backprop) can reach a very close accuracy to that of full training (with only 2-3% difference) on all datasets, namely, CIFAR-10, CIFAR-100, and TinyImageNet.

	SPARE	Random	SGD†
CIFAR-10 - ResNet-20	2.32	2.87	16.47
CIFAR-100 - ResNet-18	3.37	3.66	32.68
TinyImageNet - ResNet-50	3.05	3.51	47.43

(2.10), after some coresets updates during training. Across all datasets, we found that $\alpha = 0.1$ yielded satisfactory results.

Convergence of SPARE vs CRAIG. Figure 9.1b shows training ResNet-20 on CIFAR-10 with SPARE vs CRAIG. We see that the normalized bias of SPARE mini-batch coresets over full gradient norm, i.e., $\epsilon = \mathbb{E}[\|\boldsymbol{\xi}_{t_l}\|]/\|\nabla\mathcal{L}(\boldsymbol{w}_{t_l})\|$ is consistently small (< 1) during the training. As the gradient norm becomes smaller closer to a stationary point, small ϵ implies that the bias of the SPARE mini-batch coresets $\mathbb{E}[\|\boldsymbol{\xi}_{t_l}\|]$ diminishes as we get closer to a stationary point. Hence, convergence of SPARE can be guaranteed (Case 1 in Theorem 2.3.1). On the other hand, the normalized error for CRAIG coresets can be large during the training. Hence, convergence is not guaranteed (Case 2 in Theorem 2.3.1).

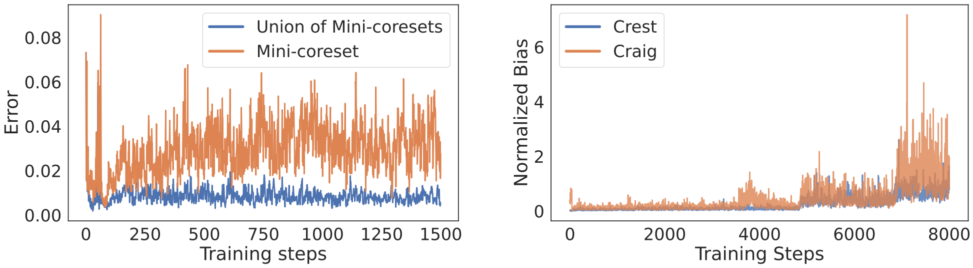
SPARE has a Similar Performance to Training with Large Mini-batches. Figure 9.4 shows the variance of gradient of SPARE mini-batch coresets of size $m = 128$ selected from random subsets V_p of size $r = 500$. We see that the variance of SPARE mini-batch coresets is very close to the variance of V_r. In contrast, random subsets of size $m = 128$ have a considerably larger variance. Figure 9.3 further compares the relative error of SPARE with mini-batch coresets of size $m = 128$ selected from random subsets of size $r = 500$. We see that training on Spare mini-batch coresets has a smaller relative error than training on random mini-batches of size $m = 128$. In particular, relative error of Spare with $m = 128$ is close to that of training on random mini-batches of size $m = 500$. This is due to the smaller gradient variance of Spare mini-batch coresets, as is shown in Figure 9.4.

Effect of Dropping the Learned Examples. By tracking the prediction accuracy of the dropped training examples (Figure 9.2a), we found that even though some of the dropped examples could be forgotten after being dropped (the accuracy of the dropped examples is 92% earlier in training), they can be learned again when training on the coresets selected from the remaining training examples (the accuracy of dropped examples always increases to above 99% even though we never train on them again). This confirms that dropping the learned examples does not harm the performance.

Spare with Larger Training Budget. In general, coreset methods are most beneficial under a limited training budget. Table 9.2 compares the relative error of training ResNet-18 on CIFAR-10, ResNet-20 on CIFAR-100 and ResNet-50 on TinyImagenet with Spare vs. Random, under 20% training budget. Note that under the standard learning rate schedule used for training on the above datasets for 200 epochs, there is a large gap up to 44.38% between SGD† (i.e., training for 20% × 200 = 40 epochs on full data with mini-batch SGD) and Spare. But, the gap reduces when learning rate drops at 60% and 85% of training (Random vs. Spare).

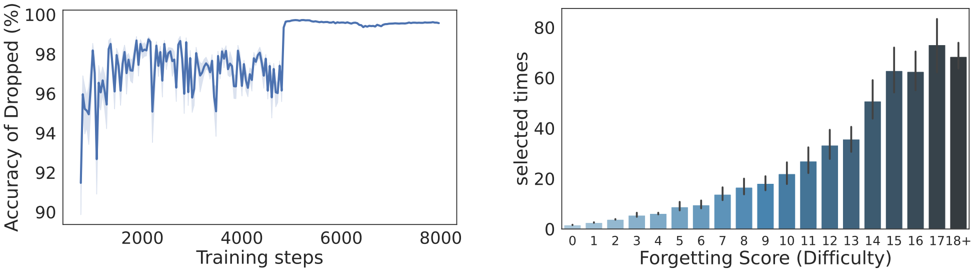
Table 9.3: Hyperparameters used for different datasets.

Dataset	τ	h
CIFAR-10	0.05	1
CIFAR-100	0.01	10
TinyImagenet	0.005	1
SNLI	0.05	4



(a) (b)

Figure 9.1: Training ResNet-20 on CIFAR-10. (a) Union of mini-batch coresets has a smaller error in capturing the full gradient, compared to the bias of the individual mini-batch coresets. (b) Normalized bias of coresets by the full gradient norm, i.e., $\epsilon = \mathbb{E}[\|\boldsymbol{\xi}_{t_l}\|]/\|\nabla \mathcal{L}(\boldsymbol{w}_{t_l})\|$ in Theorem 2.3.1. SPARE coresets have a consistently small $\epsilon < 1$. As the gradient norm becomes smaller closer to the stationary points, small ϵ implies that the bias of the SPARE mini-batch coresets $\mathbb{E}[\|\boldsymbol{\xi}_{t_l}\|]$ diminishes closer to the stationary points. Hence, convergence of SPARE can be guaranteed (Case 1 in Theorem 2.3.1). On the other hand, ϵ can be large for CRAIG coresets. Hence, convergence is not guaranteed (Case 2 in Theorem 2.3.1).



(a) (b)

Figure 9.2: Training ResNet-20 on CIFAR-10 with SPARE. (a) Dropped examples are learned later in training, by training on SPARE subsets. (b) Distribution of forgetting scores for the examples selected by SPARE during the training. The distribution is long-tailed, confirming that not all examples contribute equally to training.

Figure 9.3: Relative error (%) with 10% training budget. Training on Spare mini-batch coresets of size $m = 128$ selected from random subsets V_p of size $r = 500$ has a smaller relative error than training on random mini-batches of size $m = 128$. In particular, relative error of Spare with $m = 128$ is close to that of training on random mini-batches of size $m = 500$.

Method	Relative Error
Spare m=128	5.2
Random m=128	7.1
Random m=512	4.0

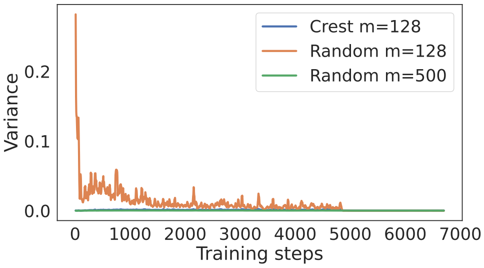


Figure 9.4: Variance of gradients of Spare mini-batches of size $m = 128$ selected from random subsets V_p of size $r = 500$ is very closer to the variance of V_r. In contrast, random subsets of size $m = 128$ have a considerably larger variance.

9.2 Appendix for Chapter 3

9.2.1 Proofs

9.2.1.1 Proof of Theorem 3.3.1

Proof. From the assumption that the loss trajectories of examples on the proxy and target models are close:

$$\|\mathbf{L}_i^{\text{proxy}} - \mathbf{L}_i^{\text{target}}\| \leq \delta, \quad \forall i. \tag{9.56}$$

Since i and j are in the same cluster C_k based on the proxy model, we have:

$$\|\mathbf{L}_i^{\text{proxy}} - \mathbf{L}_j^{\text{proxy}}\| \leq \epsilon. \tag{9.57}$$

Using the triangle inequality:

$$\|\mathbf{L}_i^{\text{target}} - \mathbf{L}_j^{\text{target}}\| \leq \|\mathbf{L}_i^{\text{target}} - \mathbf{L}_i^{\text{proxy}}\| + \|\mathbf{L}_i^{\text{proxy}} - \mathbf{L}_j^{\text{proxy}}\| + \|\mathbf{L}_j^{\text{proxy}} - \mathbf{L}_j^{\text{target}}\| \leq 2\delta + \epsilon = \epsilon'. \tag{9.58}$$

Therefore, at any iteration t:

$$|\mathcal{L}_i^{\text{target}}(\boldsymbol{\theta}^{(t)}) - \mathcal{L}_j^{\text{target}}(\boldsymbol{\theta}^{(t)})| \leq \epsilon', \quad \forall t. \tag{9.59}$$

Assuming that the loss functions can be approximated by:

$$\mathcal{L}_i^{\text{target}}(\boldsymbol{\theta}) = \frac{1}{2} d\boldsymbol{\theta}^\top \boldsymbol{H}_i d\boldsymbol{\theta} + \boldsymbol{g}_i^\top d\boldsymbol{\theta} + c_i, \tag{9.60}$$

where c_i is the loss of example i at the beginning of fine-tuning, and $d\boldsymbol{\theta}$ is the distance between the parameters of the pretrained model and those during fine-tuning. Similarly for $\mathcal{L}_j^{\text{target}}(\boldsymbol{\theta})$. The loss difference between i and j is:

$$\mathcal{L}_i^{\text{target}}(\boldsymbol{\theta}) - \mathcal{L}_j^{\text{target}}(d\boldsymbol{\theta}) = \frac{1}{2} d\boldsymbol{\theta}^\top (\boldsymbol{H}_i - \boldsymbol{H}_j) d\boldsymbol{\theta} + (\boldsymbol{g}_i - \boldsymbol{g}_j)^\top d\boldsymbol{\theta} + (c_i - c_j). \tag{9.61}$$

Given that $|\mathcal{L}_i^{\text{target}}(\boldsymbol{\theta}) - \mathcal{L}_j^{\text{target}}(\boldsymbol{\theta})| \leq \epsilon'$, we can write:

$$\left| \frac{1}{2} d\boldsymbol{\theta}^\top (\boldsymbol{H}_i - \boldsymbol{H}_j) d\boldsymbol{\theta} + (\boldsymbol{g}_i - \boldsymbol{g}_j)^\top d\boldsymbol{\theta} + (c_i - c_j) \right| \leq \epsilon'. \tag{9.62}$$

Let us choose two different values, $\boldsymbol{\theta}^{(1)}$ and $\boldsymbol{\theta}^{(2)}$, to generate two inequalities. For $d\boldsymbol{\theta}^{(1)}$, we have:

$$\left| \frac{1}{2} (d\boldsymbol{\theta}^{(1)})^\top (\boldsymbol{H}_i - \boldsymbol{H}_j) d\boldsymbol{\theta}^{(1)} + (\boldsymbol{g}_i - \boldsymbol{g}_j)^\top d\boldsymbol{\theta}^{(1)} + (c_i - c_j) \right| \leq \epsilon', \tag{9.63}$$

and for $d\boldsymbol{\theta}^{(2)}$, we have:

$$\left| \frac{1}{2} (d\boldsymbol{\theta}^{(2)})^\top (\boldsymbol{H}_i - \boldsymbol{H}_j) d\boldsymbol{\theta}^{(2)} + (\boldsymbol{g}_i - \boldsymbol{g}_j)^\top d\boldsymbol{\theta}^{(2)} + (c_i - c_j) \right| \leq \epsilon'. \tag{9.64}$$

Subtracting these two inequalities, we get:

$$\left|\frac{1}{2}\left((d\boldsymbol{\theta}^{(1)})^\top(\boldsymbol{H}_i - \boldsymbol{H}_j)\boldsymbol{\theta}^{(1)} - (d\boldsymbol{\theta}^{(2)})^\top(\boldsymbol{H}_i - \boldsymbol{H}_j)d\boldsymbol{\theta}^{(2)}\right) + (\boldsymbol{g}_i - \boldsymbol{g}_j)^\top(d\boldsymbol{\theta}^{(1)} - d\boldsymbol{\theta}^{(2)})\right| \leq 2\epsilon'. \tag{9.65}$$

$$\begin{aligned}\left|(d\boldsymbol{\theta}^{(1)})^\top(\boldsymbol{H}_i - \boldsymbol{H}_j)d\boldsymbol{\theta}^{(1)} - (d\boldsymbol{\theta}^{(2)})^\top(\boldsymbol{H}_i - \boldsymbol{H}_j)d\boldsymbol{\theta}^{(2)}\right| &\leq \|\boldsymbol{H}_i - \boldsymbol{H}_j\|\left(\|d\boldsymbol{\theta}^{(1)}\|^2 + \|d\boldsymbol{\theta}^{(2)}\|^2\right)\\ &\leq (\|\boldsymbol{H}_i\| + \|\boldsymbol{H}_j\|)\left(\|d\boldsymbol{\theta}^{(1)}\|^2 + \|d\boldsymbol{\theta}^{(2)}\|^2\right)\\ &\leq 4CD^2\end{aligned} \tag{9.66}$$

This gives us:

$$\left|(\boldsymbol{g}_i - \boldsymbol{g}_j)^\top(d\boldsymbol{\theta}^{(1)} - d\boldsymbol{\theta}^{(2)})\right| \leq 2\epsilon' + 2CD^2. \tag{9.67}$$

Assuming $\|d\boldsymbol{\theta}^{(1)} - d\boldsymbol{\theta}^{(2)}\| \geq d$, we get:

$$\|\boldsymbol{g}_i - \boldsymbol{g}_j\| \leq \frac{2\epsilon' + 2CD^2}{d} = \Delta. \tag{9.68}$$

□

9.2.1.2 Proof of Corollary 3.3.2

Without loss of generality, assume we select k example from each cluster and we have $k \leq \min_{j\in[K]} |C_j|$. Then the error of the subset in capturing the full gradient will be

$$\xi \leq \sum_j (|C_j| - k)(\bar{\boldsymbol{g}}_j + \Delta), \tag{9.69}$$

where $\bar{\boldsymbol{g}}_j$ is the norm of the average gradient of the selected examples from C_j. In practice, we can weight elements of the subset by $r_{\min}/k$, which has a similar effect to scaling the step size when training on the subset. Let $\boldsymbol{g}_{\max} = \max_j \|\boldsymbol{g}_j\|$ be the maximum gradient norm during training, $r_{\max} = \max_j |C_j|, r_{\min} = \min_j |C_j|$. Then, we get

$$\xi' \leq \sum_j (r_{\min} - k)\Delta + (|C_j| - r_{\min})(\bar{\boldsymbol{g}}_j + \Delta) \tag{9.70}$$

$$\leq K[r_{\min}\Delta + (r_{\max} - r_{\min})\boldsymbol{g}_{\max}] \tag{9.71}$$

The first term in RHS of Eq (9.70) is the error of the subset selected from C_j to capture its full gradient and the second term is due to selecting the same number of examples, k, from the larger clusters.

Using the above error and following the proof of Theorem 1 in [127], for a constant step size $\alpha \leq 1/c$ we get:

$$\|\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^*\|^2 \leq (1 - \alpha c)^{t+1}\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*\|^2 + 2\xi' R/c^2 + \alpha B^2 (r_{\min}/k)^2 \boldsymbol{g}^2_{\max}, \tag{9.72}$$

where $c \leq \|\boldsymbol{H}\|$, and $B = k \cdot K$ is the total size of the subset, $R = \min\{d_0, B\boldsymbol{g}_{\max} + \xi'/c\}$ and $d_0 = \|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^*\|$ is the initial distance to the optimal solution $\boldsymbol{\theta}^*$.

If $k \geq |C_j|$ for any cluster C_j, one can simply add $(r_{\min}/k-1)\cdot\hat{\boldsymbol{g}}_j$ to ξ' for the corresponding clusters, where $\hat{\boldsymbol{g}}_j$ is the norm of the total gradient of cluster C_j and we replace $r_{\min}$ in Eq (9.70) with the size of smallest cluster that has larger than k examples.

9.2.2 Experiment Details

9.2.2.1 Models

Pythia. The Pythia models [18] are a suite of large language models (LLMs) developed by EleutherAI licensed under the Apache License 2.0. These models range in size from 70 million to 12 billion parameters and are designed to enable controlled scientific research on transparently trained LLMs across various scales.

Phi. The Phi models [102] developed by Microsoft are under the MIT License. Phi-1.5, a transformer-based model with 1.3 billion parameters, and its successor, Phi-2, with 2.7 billion parameters, have been trained on a diverse set of data sources, including synthetic texts and curated websites. The Phi models underscore the potential of small yet powerful language models in understanding and generating human language, empowering a range of NLP tasks. Phi-2, in particular, has raised the bar for reasoning and language understanding among foundation models, matching or even exceeding the performance of models 25 times its size on complex benchmarks.

LLaMA 2. The LLaMA 2 models [185], released by Meta AI and licensed under the LLaMA 2 Community License Agreement, are designed for improved natural language understanding and generation. LLaMA 2-7B, the smallest in this series with 7 billion parameters, has demonstrated competitive performance across various NLP benchmarks despite its moderate size.

9.2.2.2 Datasets

MathInstruct. The MathInstruct dataset [224] is compiled from 13 diverse math rationale datasets, using both chain-of-thought (CoT) and program-of-thought (PoT) rationales. It ensures comprehensive coverage across various mathematical fields in the 262K training examples, making it a popular resource for fine-tuning large language models (LLMs) for general math problem-solving. MathInstruct is licensed under the MIT license.

MIMIC-III. The MIMIC-III (Medical Information Mart for Intensive Care III) dataset [83] is a comprehensive collection of de-identified health data associated with over 40,000 patients who stayed in critical care units of the Beth Israel Deaconess Medical Center in Boston, Massachusetts. This large dataset includes information such as demographics, vital signs, laboratory tests, medications, and more, making it an invaluable resource for a wide range of research in healthcare, including clinical decision support systems, medical procedure

Table 9.4: A synthetic radiology report (MRI of the brain), generated by the GPT-4 model [3] to demonstrate the typical data format and content used in the clinical text summarization task. It is not suitable for clinical or diagnostic use.

Findings	The brain parenchyma demonstrates normal morphology with no evidence of mass effect or midline shift. No acute infarcts are seen on diffusion-weighted images. There are no signs of intracranial hemorrhage. Mild generalized cerebral atrophy is noted. The ventricles and sulci appear within normal limits for the patient's age. The pituitary gland and sella turcica are unremarkable. There are no abnormal signal intensities within the brain parenchyma. The orbits, paranasal sinuses, and mastoid air cells are clear.
Impression	Normal MRI of the brain. Mild cerebral atrophy, likely age-related. No acute intracranial pathology.

efficacy studies, and patient care optimization strategies.

The MIMIC-III dataset is made freely available to the research community under the Health Insurance Portability and Accountability Act (HIPAA) compliance, ensuring patient confidentiality and data protection. Access to the dataset is granted under a data use agreement (DUA) to individuals affiliated with an institution that approves the use of the data for research purposes. Researchers seeking to utilize the MIMIC-III dataset must complete a required training course on human research protections, which ensures that all researchers are aware of the responsibilities involved in handling sensitive patient data.

9.2.2.3 Implementation Details

Spare The training trajectories for both MathInstruct and MIMIC-III are gathered from training a Pythia-70M model, the smallest model in the Pythia model suite, recorded every 500 training iterations. We utilize the Faiss library [54] to perform efficient K-means clustering of loss trajectories with Euclidean distance with $K = 100$ and 20 iterations. The hyperparameter K is tuned in the range of $\{50, 100, 200\}$ based on the average accuracy of the model trained on $30K$ selected data. We found $K = 100$ worked the best for both datasets we studied in this paper. Ablations studies on the length and the best time in the training to record the trajectories can be found in Section 5.5.2.

Comparing Reference Models for the Baselines For one-shot selection methods (excluding SPARE), we use representations from either step 1000 or the end of fine-tuning Pythia-410M on MathInstruct and reported the better result in Figure 3.4 and Table 3.1. In Table 9.5, we include the complete comparison between using early-fine-tuning vs. end-of-fine-tuning model checkpoints as the inference model. For Facility Locations, we further compared using the first hidden states as the feature representation as suggested in [16] to using the last hidden states [204] for the tasks we studied.The ranges for confidence, perplexity, and learnability are chosen according to the best-performing intervals reported in prior research (Section 3.4.1).

Due to memory and computational constraints, for Facility Locations, we calculate pairwise similarity and perform greedy selection on a per-data-source basis. We found this per-source selection approach also yields benefits for SPARE as different data sources within MathInstruct exhibit distinct common patterns in their training trajectories. Therefore, we implement SPARE also on a per-source basis for MathInstruct, and recommend applying SPARE per source when dealing with datasets composed of multiple data sources.

Hyperparameters Following the setup used in [224], we adopt a training regimen with a learning rate of 2e-5, a batch size of 128, a maximum length of 512, and a cosine scheduler with a 3% warm-up period.

Experiments Compute Resources We fine-tune all the models with the Huggingface transformers library [202] with Fully Sharded Data Parallel (FSDP) [233] on 4 48G NVIDIA RTX A6000.

Table 9.5: Complete results used for selecting the best reference model for each one-shot data selection baseline. The choice of early-fine-tuning (step 1000) and end-of-fine-tuning checkpoint follows [118]. The best results selected for Figure 3.4 are highlighted in cyan.

	Ref	Data	In-domain				Out-domain			
Selection	Model	Size	GSM8K	MATH	NumGLUE	**Avg**	SVAMP	Mathematics	SimulEq	**Avg**
Least Confidence	Early	30K	2.3	1.7	15.5	6.5	13.6	1.2	0.5	5.8
		50K	1.7	2.6	20.5	8.3	16.0	4.0	1.8	7.8
		100K	3.9	2.7	22.5	9.7	19.2	8.0	3.3	9.9
	End	30K	2.7	1.3	18.0	7.0	13.7	3.3	1.4	6.7
		50K	2.1	1.7	21.0	8.3	14.5	3.5	1.0	7.3
		100K	2.5	3.3	23.5	9.8	20.8	6.3	3.7	10.0
Middle Perplexity	Early	30K	3.3	3.8	17.5	8.2	11.8	1.2	1.2	6.5
		50K	2.9	4.1	19.6	8.9	15.6	7.6	2.9	8.8
		100K	4.8	7.1	20.4	10.8	19.6	16.1	3.9	12.0
	End	30K	5.3	3.7	16.2	8.4	14.2	8.7	1.2	8.2
		50K	3.2	5.9	20.5	9.9	18.1	11.3	**5.1**	10.7
		100K	5.4	7.2	20.9	11.2	**23.8**	15.3	3.3	12.6
High Learnability	Early	30K	6.1	1.6	19.1	8.9	10.7	9.9	1.4	8.1
		50K	6.1	2.1	18.6	8.9	14.5	14.0	2.1	8.9
		100K	**7.4**	9.2	**29.8**	**15.5**	20.7	19.4	**10.3**	**16.1**
	End	30K	3.0	1.4	14.7	6.4	2.1	6.8	1.8	5.0
		50K	1.3	2.1	16.0	6.5	4.7	6.9	3.1	5.7
		100K	4.3	7.2	23.0	11.5	16.7	16.1	4.3	11.9
Facility Location	Early (First)	50K	3.9	7.6	12.4	8.0	11.1	14.6	1.9	8.6
	Early (Last)	50K	5.7	9.1	12.4	9.1	15.4	18.6	1.6	10.5
	End (First)	50K	3.8	7.7	14.8	8.7	19.2	11.4	2.3	9.9
	End (Last)	50K	5.2	**9.7**	11.8	8.9	12.4	18.2	1.0	9.7

9.2.2.4 Evaluation

MathInstruct Datasets. We utilize 6 diverse datasets with open-formed questions for evaluating the mathematical reasoning capabilities of models trained with both the full MathInstruct dataset and selected subsets. These datasets, detailed in Table 9.6, span a range of mathematical disciplines from early algebra to calculus and linear algebra, covering various types of questions such as multi-step reasoning, arithmetic word problems, and problems from mathematics competitions. This variety ensures a comprehensive assessment across both in-domain and out-domain tasks.

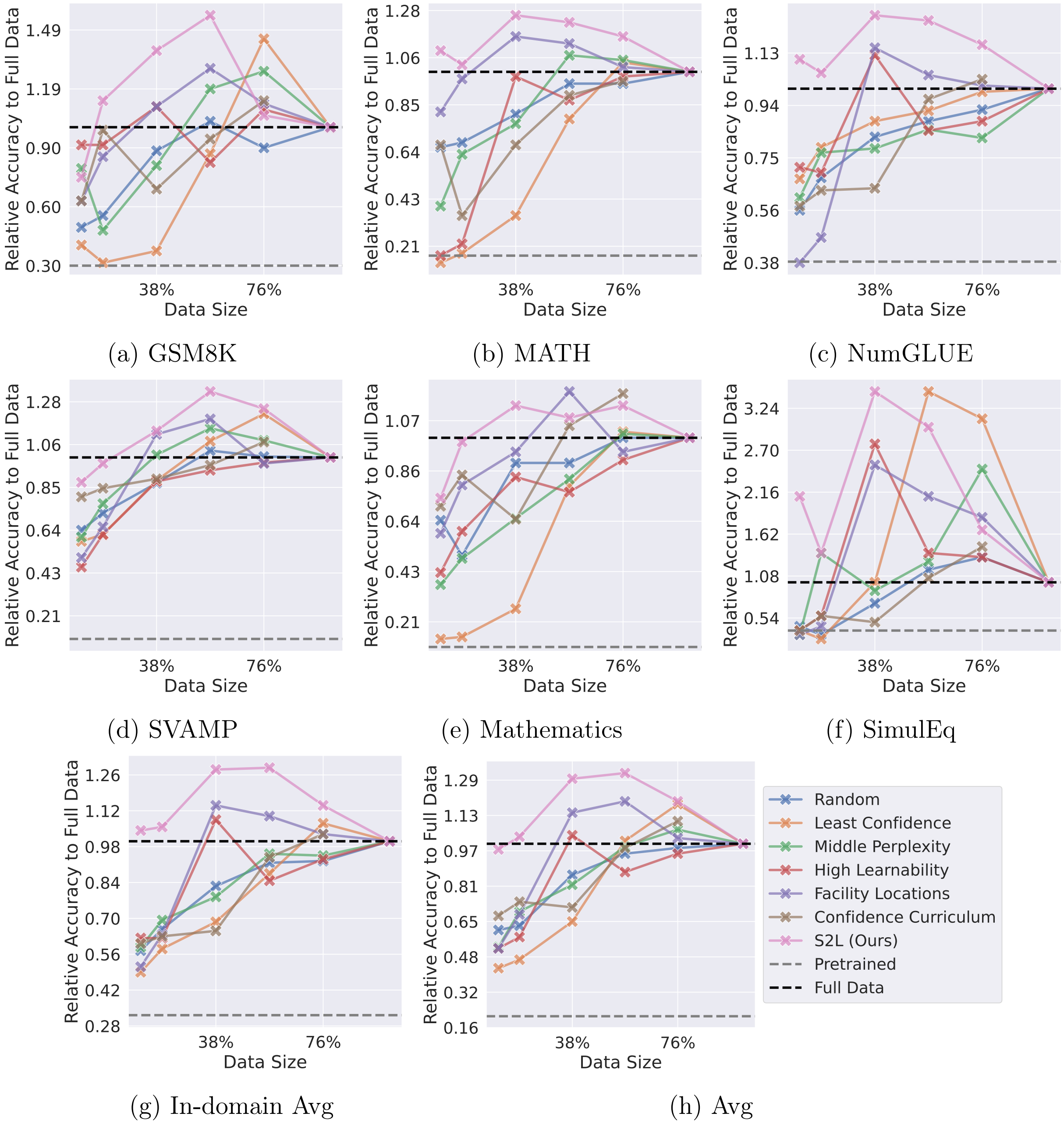
Table 9.6: Types of questions in the evaluation datasets for the mathematical reasoning task.

Dataset	Size	Level	Tasks
GSM8K	1319	Early Algebra	Multi-step reasoning using basic arithmetic operations
MATH	5000	Early Algebra, Intermediate Algebra, Algebra, Probability, NumTheory, Calculus, Geometry	Problems from mathematics competitions, including the AMC 10, AMC 12, AIME
NumGLUE	1042	Early Algebra	Commonsense, Domain-specific, Arithmetic Reasoning, Quantitative Comparison, Fill-in-the-blanks Format, Reading Comprehension, Numerical Reasoning, Quantitative NLI, Arithmetic Word Problems
SVAMP	1000	Early Algebra	Arithmetic Word Problems
Mathematics	1000	Early Algebra, Intermediate Algebra, NumTheory, Calculus	Arithmetic Reasoning
SimulEq	514	Linear Algebra	Single and multiple equation word problems

MathInstruct Pipeline. We utilize the pipeline provided by [224][1], designed to first determine whether the model can be prompted to generate a code snippet. This code snippet, if successfully generated, should be executable and produce the correct answer when run. This code-based evaluation is also used for Phi models [102]. In cases where the model does not directly produce a viable code solution, we employ a “think step-by-step" prompting strategy [199]. This method prompts the model to break down its reasoning process, a technique that has been widely proven effective in fully exploiting the model’s problem-solving capacity.

MIMIC-III. Following [48, 50], we include the six most common modality/anatomy pairs: CT head, CT abdomen, CT chest, MRI head, CT spine, and CT neck, and five less common pairs in the text data: MRI spine, CT sinus, MRI abdomen, MRI pelvis, and MRI neck in the evaluation. There are in total 13.7K test examples after data preprocessing and train-test splitting.

[1] https://github.com/TIGER-AI-Lab/MAmmoTH?tab=readme-ov-file#large-scale-evaluation



(a) GSM8K

(b) MATH

(c) NumGLUE

(d) SVAMP

(e) Mathematics

(f) SimulEq

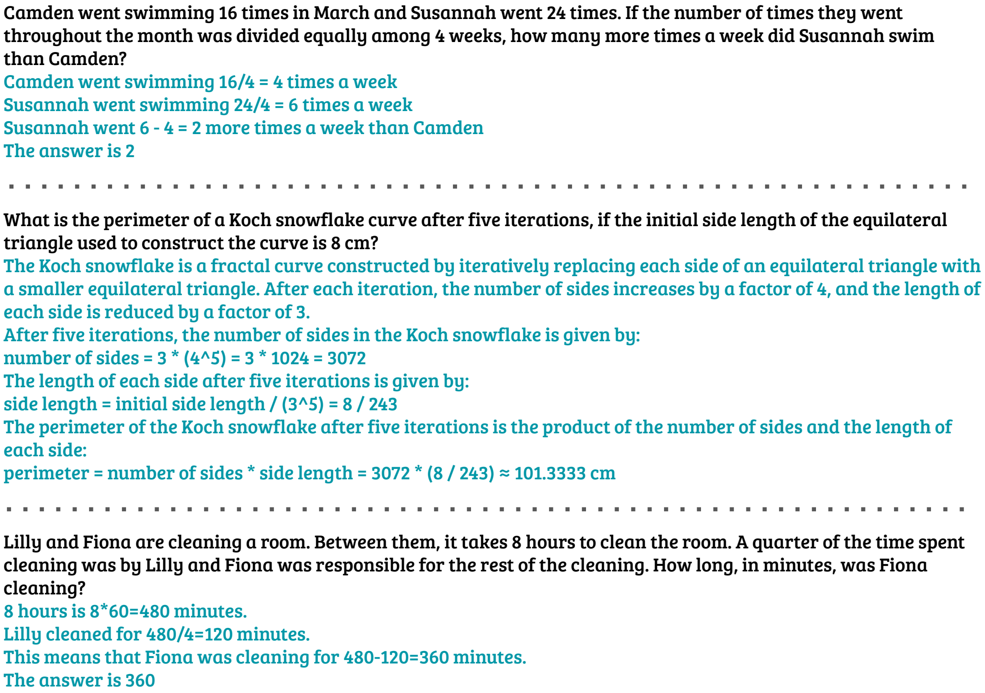
(g) In-domain Avg

(h) Avg

Figure 9.5: Break-down accuracies (↑) on in-domain and out-of-domain datasets using Pythia-410M. Data size refers to the total number of unique training examples used. All experiments in this table share the same total training steps and learning rate schedule (see Section 3.4.2).

9.2.3 Examples in Different Clusters

We compare data points in the same and different clusters based on training trajectories, in Figure 9.6, Figure 9.7 and Figure 9.8. We can observe that examples with similar training trajectories have the same question format. Therefore, balanced sampling from all clusters can ensure different types of examples can be covered in the selected subset of training data.



Camden went swimming 16 times in March and Susannah went 24 times. If the number of times they went throughout the month was divided equally among 4 weeks, how many more times a week did Susannah swim than Camden?

Camden went swimming 16/4 = 4 times a week
Susannah went swimming 24/4 = 6 times a week
Susannah went 6 - 4 = 2 more times a week than Camden
The answer is 2

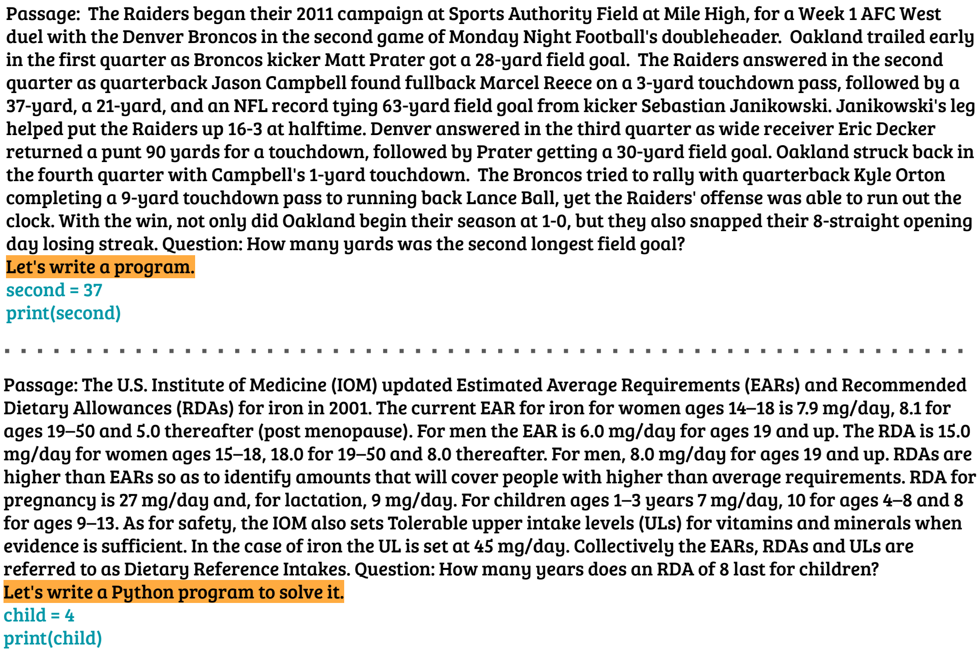
What is the perimeter of a Koch snowflake curve after five iterations, if the initial side length of the equilateral triangle used to construct the curve is 8 cm?

The Koch snowflake is a fractal curve constructed by iteratively replacing each side of an equilateral triangle with a smaller equilateral triangle. After each iteration, the number of sides increases by a factor of 4, and the length of each side is reduced by a factor of 3.
After five iterations, the number of sides in the Koch snowflake is given by:
number of sides = 3 * (4^5) = 3 * 1024 = 3072
The length of each side after five iterations is given by:
side length = initial side length / (3^5) = 8 / 243
The perimeter of the Koch snowflake after five iterations is the product of the number of sides and the length of each side:
perimeter = number of sides * side length = 3072 * (8 / 243) ≈ 101.3333 cm

Lilly and Fiona are cleaning a room. Between them, it takes 8 hours to clean the room. A quarter of the time spent cleaning was by Lilly and Fiona was responsible for the rest of the cleaning. How long, in minutes, was Fiona cleaning?

8 hours is 8*60=480 minutes.
Lilly cleaned for 480/4=120 minutes.
This means that Fiona was cleaning for 480-120=360 minutes.
The answer is 360

Figure 9.6: Examples in the cluster shown in Figure 3.7a: open-formed algebra. Questions are in black and answers are in cyan.



Passage: The Raiders began their 2011 campaign at Sports Authority Field at Mile High, for a Week 1 AFC West duel with the Denver Broncos in the second game of Monday Night Football's doubleheader. Oakland trailed early in the first quarter as Broncos kicker Matt Prater got a 28-yard field goal. The Raiders answered in the second quarter as quarterback Jason Campbell found fullback Marcel Reece on a 3-yard touchdown pass, followed by a 37-yard, a 21-yard, and an NFL record tying 63-yard field goal from kicker Sebastian Janikowski. Janikowski's leg helped put the Raiders up 16-3 at halftime. Denver answered in the third quarter as wide receiver Eric Decker returned a punt 90 yards for a touchdown, followed by Prater getting a 30-yard field goal. Oakland struck back in the fourth quarter with Campbell's 1-yard touchdown. The Broncos tried to rally with quarterback Kyle Orton completing a 9-yard touchdown pass to running back Lance Ball, yet the Raiders' offense was able to run out the clock. With the win, not only did Oakland begin their season at 1-0, but they also snapped their 8-straight opening day losing streak. Question: How many yards was the second longest field goal?
Let's write a program.

```
second = 37
print(second)
```

Passage: The U.S. Institute of Medicine (IOM) updated Estimated Average Requirements (EARs) and Recommended Dietary Allowances (RDAs) for iron in 2001. The current EAR for iron for women ages 14–18 is 7.9 mg/day, 8.1 for ages 19–50 and 5.0 thereafter (post menopause). For men the EAR is 6.0 mg/day for ages 19 and up. The RDA is 15.0 mg/day for women ages 15–18, 18.0 for 19–50 and 8.0 thereafter. For men, 8.0 mg/day for ages 19 and up. RDAs are higher than EARs so as to identify amounts that will cover people with higher than average requirements. RDA for pregnancy is 27 mg/day and, for lactation, 9 mg/day. For children ages 1–3 years 7 mg/day, 10 for ages 4–8 and 8 for ages 9–13. As for safety, the IOM also sets Tolerable upper intake levels (ULs) for vitamins and minerals when evidence is sufficient. In the case of iron the UL is set at 45 mg/day. Collectively the EARs, RDAs and ULs are referred to as Dietary Reference Intakes. Question: How many years does an RDA of 8 last for children?
Let's write a Python program to solve it.

```
child = 4
print(child)
```

Figure 9.7: Examples in the cluster shown in Figure 3.7b: reading comprehension + coding. Questions are in black and answers are in cyan; instructions are highlighted in orange.

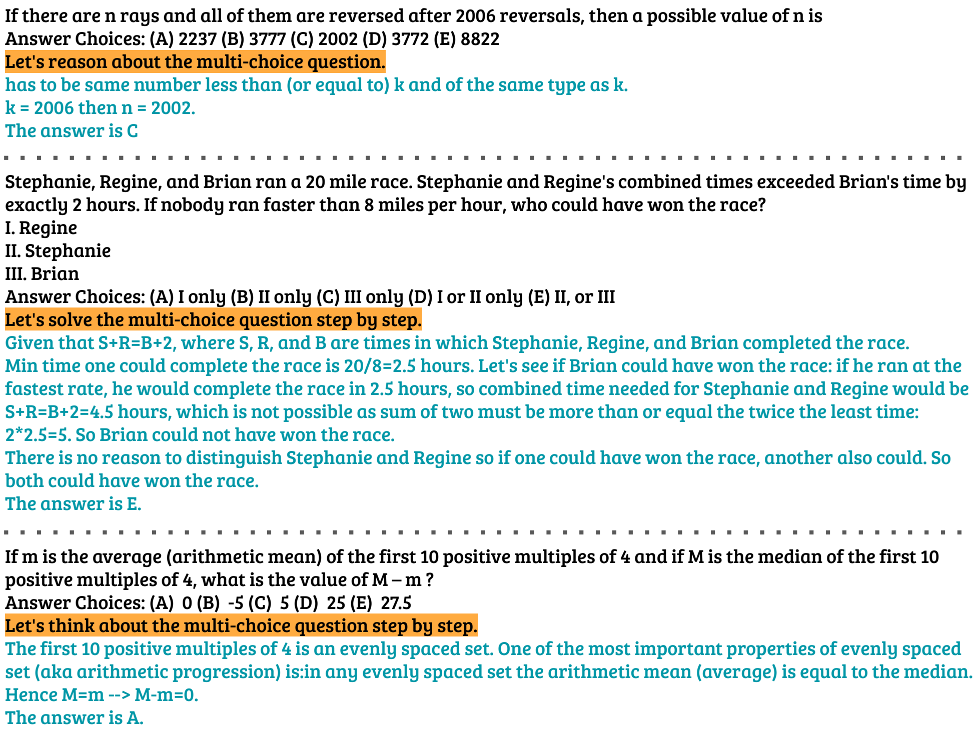
9.2.4 Topic Distribution of Data Selected by Spare

Beyond qualitative examples from different clusters, we study how Spare changes the data distribution to outperform using the full fine-tuning dataset as well as using random subsets of the same size that have the same distribution as the original dataset. In Figure 9.9, we can observe that Spare not only guarantees a thorough and balanced coverage across the spectrum of topics but also ensures sufficient representation of foundational topics, such as pre-algebra, which lays the groundwork for tackling more complex subjects.

9.2.5 Broader Impacts

This paper introduces a data selection method for large language models (LLMs), aiming to enhance the data efficiency in the supervised fine-tuning (SFT) of these models.

Positive Impacts: Our method, by reducing the data requirements for training LLMs, can make fine-tuning LLMs more effective and accessible. This could lead to broader



If there are n rays and all of them are reversed after 2006 reversals, then a possible value of n is
Answer Choices: (A) 2237 (B) 3777 (C) 2002 (D) 3772 (E) 8822
Let's reason about the multi-choice question.
has to be same number less than (or equal to) k and of the same type as k.
k = 2006 then n = 2002.
The answer is C

Stephanie, Regine, and Brian ran a 20 mile race. Stephanie and Regine's combined times exceeded Brian's time by exactly 2 hours. If nobody ran faster than 8 miles per hour, who could have won the race?
I. Regine
II. Stephanie
III. Brian
Answer Choices: (A) I only (B) II only (C) III only (D) I or II only (E) II, or III
Let's solve the multi-choice question step by step.
Given that S+R=B+2, where S, R, and B are times in which Stephanie, Regine, and Brian completed the race.
Min time one could complete the race is 20/8=2.5 hours. Let's see if Brian could have won the race: if he ran at the fastest rate, he would complete the race in 2.5 hours, so combined time needed for Stephanie and Regine would be S+R=B+2=4.5 hours, which is not possible as sum of two must be more than or equal the twice the least time: 2*2.5=5. So Brian could not have won the race.
There is no reason to distinguish Stephanie and Regine so if one could have won the race, another also could. So both could have won the race.
The answer is E.

If m is the average (arithmetic mean) of the first 10 positive multiples of 4 and if M is the median of the first 10 positive multiples of 4, what is the value of M – m ?
Answer Choices: (A) 0 (B) -5 (C) 5 (D) 25 (E) 27.5
Let's think about the multi-choice question step by step.
The first 10 positive multiples of 4 is an evenly spaced set. One of the most important properties of evenly spaced set (aka arithmetic progression) is:in any evenly spaced set the arithmetic mean (average) is equal to the median.
Hence M=m --> M-m=0.
The answer is A.

Figure 9.8: Examples in the cluster shown in Figure 3.7c: multiple-choice + multi-step reasoning. Questions are in black and answers are in cyan; instructions are highlighted in orange.

participation in AI research and application development across various fields, including healthcare and education.

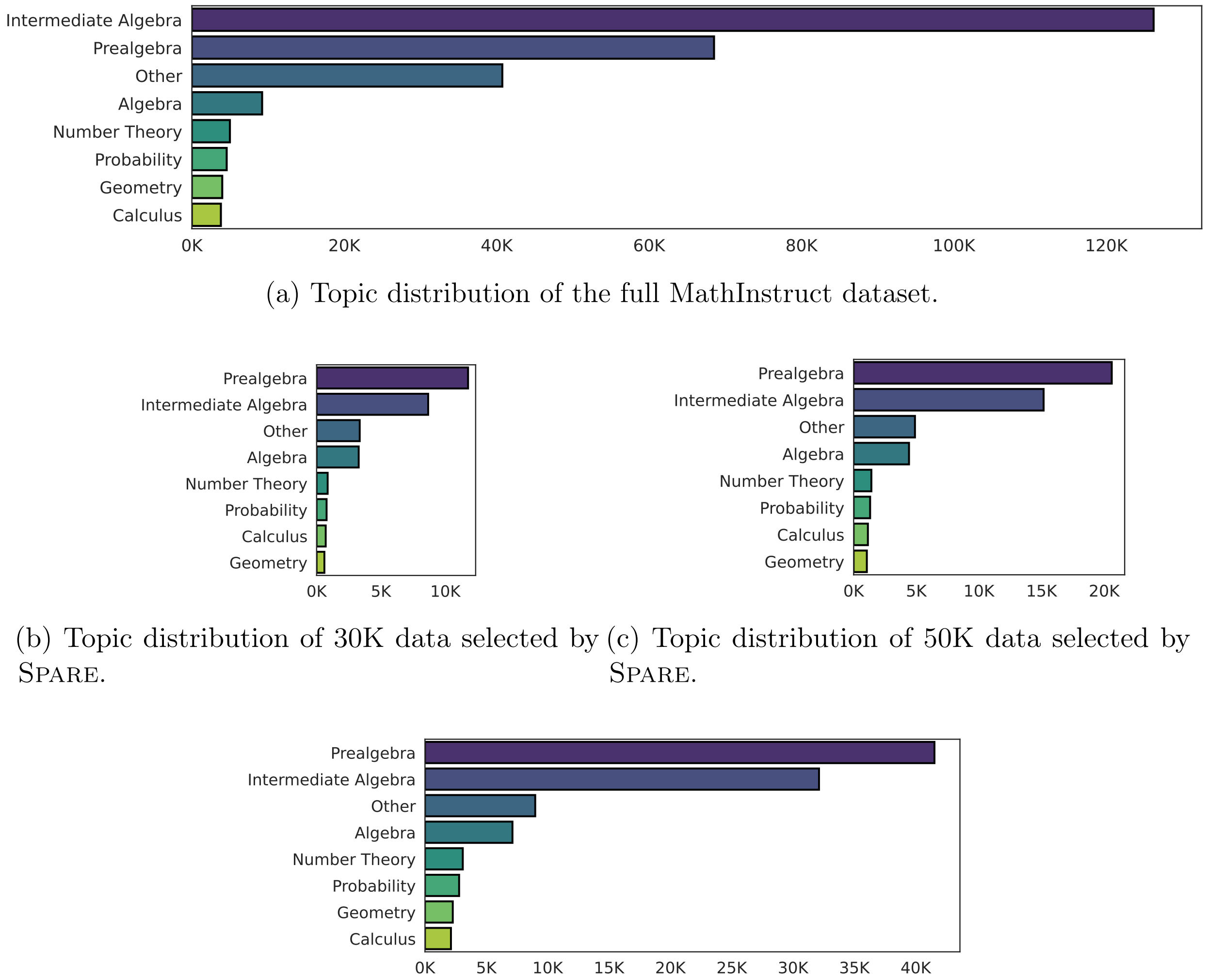
Negative Impacts: Our method does not inherently involve or encourage applications with direct negative societal impacts. The focus is on a generic improvement in the field of machine learning, particularly in the training of LLMs.

9.3 Appendix for Chapter 4

9.3.1 Proof of Theorem 4.2.1

A loss function $\mathcal{L}(w)$ is considered μ -PL on a set $\mathcal{S}$, if the following holds:

$$\frac{1}{2}\|\mathbf{g}\|^2 \geq \mu\left(\mathcal{L}(w)-\mathcal{L}\left(w_*\right)\right), \forall w \in \mathcal{S} \tag{9.73}$$



(a) Topic distribution of the full MathInstruct dataset.

(b) Topic distribution of 30K data selected by SPARE.

(c) Topic distribution of 50K data selected by SPARE.

(d) Topic distribution of 100K data selected by SPARE.

Figure 9.9: Compared to the original topic distribution, SPARE prioritized easier topics (e.g., pre-algebra over intermediate algebra, algebra over other more advanced topics) while always ensuring complete and more balanced coverage of all topics.

where w_* is a global minimizer. When additionally $\mathcal{L}(w_*) = 0$, the μ-PL condition is equivalent to the μ-PL* condition

$$\frac{1}{2}\|\mathbf{g}\|^2 \geq \mu\mathcal{L}(w), \forall w \in \mathcal{S}. \tag{9.74}$$

For Lipschitz continuous $\mathbf{g}$ and μ-PL* condition, gradient descent on the entire dataset yields

$$\mathcal{L}(w_{t+1}) - \mathcal{L}(w_t) \leq -\frac{\eta}{2}\|\mathbf{g}_t\|^2 \leq -\eta\mu\mathcal{L}(w_t), \tag{9.75}$$

and,

$$\mathcal{L}(w_t) \leq (1-\eta\mu)^t\mathcal{L}(w_0), \tag{9.76}$$

which was shown in [105]. We build upon this result.

For the subset we have

$$\mathcal{L}(w_{t+1}) - \mathcal{L}(w_t) \leq -\frac{\eta}{2}\|\mathbf{g}_t^S\|^2 \tag{9.77}$$

By substituting Eq. (9.75) we have.

$$\begin{align}
&\leq -\frac{\eta}{2}(\|\mathbf{g}_t\| - \rho)^2 \tag{9.78}\\
&= -\frac{\eta}{2}(\|\mathbf{g}_t\|^2 + \rho^2 - 2\rho\|\mathbf{g}_t\|) \tag{9.79}\\
&\leq -\frac{\eta}{2}(\|\mathbf{g}_t\|^2 + \rho^2 - 2\rho\nabla_{\max}) \tag{9.80}\\
&\leq -\frac{\eta}{2}(2\mu\mathcal{L}(w_t) + \rho^2 - 2\rho\nabla_{\max}) \tag{9.81}
\end{align}$$

where we can upper bound the norm of $\mathbf{g}_t$ in Eq. (9.79) by a constant $\nabla_{\max}$. And Eq. (9.81) follows from the μ-PL condition from Eq. (9.73). While the loss is very non-convex during the first part of training, it becomes nearly convex afterward [58]. SPARE starts dropping points after a few epochs of training, where Lipschitzness, μ-PL condition, and norm-bounded gradients are likely to hold. In Eq. (9.76), LHS directly results from Lipschitzness [24].

Hence,

$$\mathcal{L}(w_{t+1}) \leq (1-\eta\mu)\mathcal{L}(w_t) - \frac{\eta}{2}(\rho^2 - 2\rho\nabla_{\max}) \tag{9.82}$$

Since, $\sum_{j=0}^{k}(1-\eta\mu)^j \leq \frac{1}{\eta\mu}$, for a constant learning rate η we get

$$\mathcal{L}(w_{t+1}) \leq (1-\eta\mu)^{t+1}\mathcal{L}(w_0) - \frac{1}{2\mu}(\rho^2 - 2\rho\nabla_{\max}) \tag{9.83}$$

9.4 Appendix for Chapter 5

9.4.1 Simplicity Bias

A recent body of work revealed that the neural network trained with (stochastic) gradient methods can be approximated on the training data by a linear function early in training [69, 71, 131, 136, 146, 164]. We hypothesize that a slightly stronger statement holds, namely the approximation still holds if we isolate a core or spurious feature from an example and input it to the model.

Assumption 9.4.1 (simplicity bias on core and spurious features, informal)**.** Suppose that f^{lin} is a linear function that closely approximates $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{z})$ on the training data. Then f^{lin} also approximates f on input either a core feature or a spurious feature corresponding to a majority group in some class, that is

$$\begin{aligned} f^{lin}(\boldsymbol{v}_c) &\approx f(\boldsymbol{v}_c; \boldsymbol{W}, \boldsymbol{z}) \qquad \forall c \in \mathcal{C} \\ f^{lin}(\boldsymbol{v}_s) &\approx f(\boldsymbol{v}_s; \boldsymbol{W}, \boldsymbol{z}) \qquad \forall s \in \mathcal{A} \end{aligned}$$

Intuitively, every core feature and every spurious feature corresponding to a majority group is well represented in the training dataset, and since it is known that the linear model and the full neural network agree on the training dataset, we can expect them to agree on such features as well. Note that spurious features that do not appear in majority groups may not be well represented in the training dataset, hence we do not require that the linear model approximates the neural network well on such features. The formal statement is provided below as Assumption 9.4.6.

9.4.2 Setting

We now introduce the formal mathematical setting for the theory.

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$, be a dataset with covariance $\boldsymbol{\Sigma}$. Define the data matrix

$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_n \end{bmatrix}^\top$ and the label vector $\boldsymbol{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}^\top$. We use $\| \cdot \|$ to refer to the Euclidean norm of a vector or the spectral norm of the data.

Following [71], we make the following assumptions:

Assumption 9.4.2 (input distribution)**.** The data has the following properties (with high probability):

$$\begin{aligned} \frac{\|\boldsymbol{x}_i\|^2}{d} &= 1 \pm O(\sqrt{\frac{\log n}{d}}), \forall i \in [n] \\ \frac{|\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle|}{d} &= O(\sqrt{\frac{\log n}{d}}), \forall i, j \in [n], i \neq j \\ \|\boldsymbol{X}\boldsymbol{X}^\top\| &= \Theta(n) \end{aligned}$$

Assumption 9.4.3 (activation function)**.** The activation $\phi(\cdot)$ satisfies either of the following:

- smooth activation: ϕ has bounded first and second derivative
- piecewise linear activation:

$$\phi(z) = \begin{cases} z & z \geq 0 \\ az & z < 0 \end{cases}$$

Assumption 9.4.4 (initialization)**.** The weights $(\boldsymbol{W}, \boldsymbol{v})$ are initialized using symmetric initialization:

$$\begin{aligned} \boldsymbol{w}_1, \dots, \boldsymbol{w}_{\frac{m}{2}} \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d), \qquad & \boldsymbol{w}_{i+\frac{m}{2}} = \boldsymbol{w}_i (\forall i \in 1, \dots, \frac{m}{2}) \\ v_1, \dots, v_{\frac{m}{2}} \sim \text{Unif}(\{-1, 1\}), \qquad & v_{i+\frac{m}{2}} = -v_i (\forall i \in 1, \dots, \frac{m}{2}) \end{aligned}$$

It is not hard to check that the concrete scenario we choose in our analysis satisfies the above assumptions. Now, given the following assumptions, we leverage the result of [71]:

Theorem 9.4.5 ([71])**.** *Let $\alpha \in (0, 1/4)$ be a fixed constant. Suppose d is the input dimensionality, $\frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle}{d} = \mathbb{1}_{i=j} \pm O(\sqrt{\frac{\log n}{d}}), \forall i, j \in [n]$, the data matrix $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ has spectral norm $\left\|\boldsymbol{X}\boldsymbol{X}^\top\right\| = \Theta(n)$, and for the labels we have $|y_i| \leq 1$ $\forall y_i$. Assume the number of training* *samples n and the network width m satisfy $n, m = \Omega(d^{1+\alpha}), n, m \leq d^{\mathcal{O}(1)}$, and the learning rate $\eta \ll d$. Then, there exist a universal constant C, such that with high probability for all $0 \leq t \leq T = C \cdot \frac{d \log d}{\eta}$, the network $f(\boldsymbol{w}_t, \boldsymbol{X})$ trained with GD is very close to a linear function $f^{lin}(\boldsymbol{\beta}, \boldsymbol{X})$:*

$$\frac{1}{n}\sum_{i=1}^{n}(f^{lin}(\boldsymbol{\beta}_t, \boldsymbol{X}) - f(\boldsymbol{w}_t, \boldsymbol{X}))^2 \leq \frac{\eta^2 t^2}{d^{2+\Omega(\alpha)}} \leq \frac{1}{d^{\Omega(\alpha)}}. \tag{9.84}$$

In particular, the linear model $f^{lin}(\boldsymbol{\beta}, \boldsymbol{X})$ takes operates on the transformed data $\boldsymbol{\psi}(\boldsymbol{x})$, where

$$\begin{aligned}
\boldsymbol{\psi}(\boldsymbol{x}) &= \begin{bmatrix} \sqrt{\frac{2}{d}}\zeta \boldsymbol{x} \\ \sqrt{\frac{3}{2d}}\nu \\ \vartheta_0 + \vartheta_1(\frac{\|\boldsymbol{x}\|}{\sqrt{d}} - 1) + \vartheta_2(\frac{\|\boldsymbol{x}\|}{\sqrt{d}} - 1)^2 \end{bmatrix} \\
\zeta &= \mathbb{E}_{g\sim\mathcal{N}(0,1)}[\phi'(g)] \\
\nu &= \mathbb{E}_{g\sim\mathcal{N}(0,1)}[g\phi'(g)]\sqrt{\frac{\mathrm{Tr}[\boldsymbol{\Sigma}^2]}{d}} \\
\vartheta_0 &= \mathbb{E}_{g\sim\mathcal{N}(0,1)}[g] \\
\vartheta_1 &= \mathbb{E}_{g\sim\mathcal{N}(0,1)}[g\phi'(g)] \\
\vartheta_2 &= \mathbb{E}_{g\sim\mathcal{N}(0,1)}[(\frac{1}{2}g^3 - g)\phi'(g)]
\end{aligned}$$

Note that $\boldsymbol{\psi}(\boldsymbol{x})$ consists of a scaled version of the data, a bias term, and a term that depends on the norm of the example. We will adopt the notation $f(\boldsymbol{\psi}; \boldsymbol{\beta}) = \boldsymbol{\psi}^\top \boldsymbol{\beta}$ for the linear model.

We can now formally state 9.4.1:

Assumption 9.4.6 (formal version of 9.4.1)**.** Suppose that Theorem 9.4.5 holds. Then with high probability, for all such t the following also holds for all $c \in \mathcal{C}$ and for all $s \in \mathcal{A}$:

$$\begin{aligned}
|f^{lin}(\boldsymbol{\beta}_t, \boldsymbol{v}_c) - f(\boldsymbol{w}_t, \boldsymbol{v}_c)| &\leq \frac{\eta t}{d^{1+\Omega(\alpha)}}, \\
|f^{lin}(\boldsymbol{\beta}_t, \boldsymbol{v}_s) - f(\boldsymbol{w}_t, \boldsymbol{v}_s)| &\leq \frac{\eta t}{d^{1+\Omega(\alpha)}}.
\end{aligned}$$

We will assume the former holds in the proof of the following theorems, although as we will see the assumption is unnecessary for Theorem 5.3.2.

9.4.3 Proof for Theorems

9.4.3.1 Notation

For the analysis, we split $\boldsymbol{\beta}$ into its components corresponding to the data, bias and norm parts of $\boldsymbol{\psi}$; that is $\boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{\beta}' \\ \beta_{bias} \\ \beta_{norm} \end{pmatrix}$ for $\boldsymbol{\beta}' \in \mathbb{R}^d, \beta_{bias} \in \mathbb{R}, \beta_{norm} \in \mathbb{R}$. We use the inner product between $\boldsymbol{\beta}'$ and a feature $\boldsymbol{v}$ to understand how well the linear model learns a feature $\boldsymbol{v} \in \mathbb{R}^d$. With slight abuse of notation, we will simply write $\langle \boldsymbol{\beta}, \boldsymbol{v} \rangle$ to mean $\langle \boldsymbol{\beta}', \boldsymbol{v} \rangle$.

We also define the matrix $\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1 & \dots & \boldsymbol{\phi}_n \end{bmatrix}^\top$.

9.4.3.2 Proof of Theorem 5.3.1 and 5.3.2

Theorem 5.3.1. *Let $\alpha \in (0, \frac{1}{4})$ be a fixed constant. Suppose the number of training samples n and the network width m satisfy $n \gtrsim d^{1+\alpha}$ and $m \gtrsim d^{1+\alpha}$. Let n_c be the number of examples in class c, and $n_{c,s} = |g_{c,s}|$ be the size of group $g_{c,s}$ with label c and spurious feature $\boldsymbol{v}_s \in \mathcal{A}$. Then, under the setting of Sec. 5.2 there exist a constant $\nu_1 > 0$, such that with high probability, for all $0 \le t \le \nu_1 \cdot \sqrt{\frac{d^{1-\alpha}}{\eta}}$, the contribution of the core and spurious features to the network output can be quantified as follows:*

$$f(\boldsymbol{v}_c; \boldsymbol{W}_t, \boldsymbol{z}_t) = \sqrt{\frac{2}{d}}\eta\zeta c\|\boldsymbol{v}_c\|^2 t \left(\frac{n_c}{n} \pm \mathcal{O}(d^{-\Omega(\alpha)})\right), \tag{5.5}$$

$$f(\boldsymbol{v}_s; \boldsymbol{W}_t, \boldsymbol{z}_t) = \sqrt{\frac{2}{d}}\eta\zeta c\|\boldsymbol{v}_s\|^2 t \left(\frac{n_{c,s} - n_{c',s}}{n} \pm \mathcal{O}(d^{-\Omega(\alpha)})\right), \tag{5.6}$$

where $c' = \mathcal{C} \backslash c$, and ζ is the expected gradient of activation functions at random initialization.

Corollary 5.3.2 (Separability of majority and minority groups). *Suppose that for all classes, a majority group has at least K examples and a minority group has at most k*

examples. Then, under the assumptions of Theorem 5.3.1, examples in the majority and minority groups are separable based on the model's output, early in training. That is, for all $0 \leq t \leq \nu_1 \cdot \sqrt{\frac{d^{1-\alpha}}{\eta}}$*, with high probability, the following holds for at least* $1 - \mathcal{O}(d^{-\Omega(\alpha)})$ *fraction of the training examples* $\boldsymbol{x}_i$ *in group* $g_{c,s}$*:*

If $g_{c,s}$ *is in a majority group in class* $c = 1$*:*

$$f(\boldsymbol{x}_i; \boldsymbol{W}_t, \boldsymbol{z}_t) \geq \frac{2\eta\zeta^2 t}{d}\left(\frac{\|\boldsymbol{v}_s\|^2(K-k)}{n} + \xi \pm O(d^{-\Omega(\alpha)})\right) + \rho(t, \phi, \Sigma), \tag{5.7}$$

If $g_{c,s}$ *is in a minority group in class* $c = 1$*, but* $g_{c',s}$ *is a majority group in class* $c' = -1$*:*

$$f(\boldsymbol{x}_i; \boldsymbol{W}_t, \boldsymbol{z}_t) \leq \frac{2\eta\zeta^2 t}{d}\left(-\frac{\|\boldsymbol{v}_s\|^2(K-k)}{n} + \xi \pm O(d^{-\Omega(\alpha)})\right) + \rho(t, \phi, \Sigma), \tag{5.8}$$

where ρ *is constant for all examples in the same class,* $\xi \sim \mathcal{N}(0, \kappa)$ *with*

$$\kappa = \frac{1}{n}(\sum_c n_c^2 \sigma_c^2 \|\boldsymbol{v}_c\|^2)^{1/2} + \frac{1}{n}(\sum_s (n_{c,s} - n_{c',s})^2 \sigma_s^2 \|\boldsymbol{v}_s\|^2)^{1/2}$$

is the total effect of noise on the model.

Analogous statements holds for the class $c = -1$ *by changing the sign and direction of the inequality.*

As in [71], we will conduct our analysis under the high probability events that $\|\boldsymbol{\Psi}^\top \boldsymbol{\Psi}\| = O(\frac{n}{d})$ and for all training data $\boldsymbol{x}$, $\frac{\|\boldsymbol{x}\|}{\sqrt{d}} = 1 \pm O(\sqrt{\frac{\log n}{d}})$.

Starting from the rule of gradient descent

$$\begin{aligned}\boldsymbol{\beta}(t+1) &= \boldsymbol{\beta}(t) - \frac{\eta}{n}\boldsymbol{\Psi}^\top(\boldsymbol{\Psi}\boldsymbol{\beta}(t) - \boldsymbol{y}) \\ &= \left(I - \frac{\eta}{n}\boldsymbol{\Psi}^\top\boldsymbol{\Psi}\right)\boldsymbol{\beta}(t) + \frac{\eta}{n}\boldsymbol{\Psi}^\top\boldsymbol{y}\end{aligned}$$

Let $\boldsymbol{A} = \boldsymbol{I} - \frac{\eta}{n}\boldsymbol{\Psi}^\top\boldsymbol{\Psi}, \boldsymbol{b} = \frac{\eta}{n}\boldsymbol{\Psi}^\top\boldsymbol{y}$. Also, $\boldsymbol{A}$ can be diagonalized as $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top$. Since $\|\boldsymbol{\Psi}^\top\boldsymbol{\Psi}\| = O(\frac{n}{d})$, the eigenvalues of $\boldsymbol{A}$, call them $\lambda_1, \ldots, \lambda_d$, are of order $1 - O(\frac{\eta}{d})$. For $t \geq 1$, the previous recurrence relation admits the solution

$$\begin{aligned}\boldsymbol{\beta}(t) &= (\boldsymbol{I} + \boldsymbol{A} + \cdots + \boldsymbol{A}^{t-1})\boldsymbol{b} \\ &= \boldsymbol{V}(\boldsymbol{I} + \boldsymbol{D} + \cdots + \boldsymbol{D}^{t-1})\boldsymbol{V}^\top \boldsymbol{b}\end{aligned}$$

When $t = O(\sqrt{\frac{d^{1-\alpha}}{\eta}})$, the eigenvalues of $\boldsymbol{I} + \boldsymbol{D} + \cdots + \boldsymbol{D}^{t-1}$ are on the order of

$$\begin{aligned}1 + \lambda_i + \cdots + \lambda_i^{t-1} &= \frac{1 - \lambda_i^t}{1 - \lambda_i} \\ &= 1 + O(d^{-\frac{\alpha}{2}})\end{aligned}$$

Thus we can approximate $\boldsymbol{I} + \boldsymbol{D} + \cdots + \boldsymbol{D}^{t-1} = t\boldsymbol{I} + \boldsymbol{\Delta}$, where $\|\boldsymbol{\Delta}\| = O(d^{-\frac{\alpha}{2}})$. Then

$$\beta(t) = \boldsymbol{V}(t\boldsymbol{I} + \boldsymbol{\Delta})\boldsymbol{V}^\top \boldsymbol{b} = t\boldsymbol{b} + \boldsymbol{\Delta}_1 \boldsymbol{b}$$

where $\boldsymbol{\Delta}_1 = \boldsymbol{V}\boldsymbol{\Delta}\boldsymbol{V}^\top$ also satisfies $\|\boldsymbol{\Delta}\| = O(d^{-\frac{\alpha}{2}})$.

From here we may calculate the following: the alignment of $\boldsymbol{\beta}$ with a core feature $\boldsymbol{v}_c$ is

$$\begin{aligned}\langle \boldsymbol{v}_c, \boldsymbol{\beta} \rangle &= \sqrt{\frac{2}{d}} \frac{\eta\zeta c \|\boldsymbol{v}_c\|}{n} (t \pm O(d^{-\frac{\alpha}{2}}))(\|\boldsymbol{v}_c\| n_c \pm O(\sigma_c \sqrt{n})) \quad (9.85) \\ &= \sqrt{\frac{2}{d}} \eta\zeta c \|\boldsymbol{v}_c\|^2 t \left(\frac{n_c}{n} \pm O(d^{-\Omega(\alpha)}) \right) \quad (9.86)\end{aligned}$$

and the alignment with a spurious feature $\boldsymbol{v}_s$ is

$$\begin{aligned}\langle \boldsymbol{v}_s, \boldsymbol{\beta} \rangle &= \sqrt{\frac{2}{d}} \frac{\eta\zeta c \|\boldsymbol{v}_s\|}{n} (t \pm O(d^{-\frac{\alpha}{2}}))(\|\boldsymbol{v}_s\| (n_{c,s} - n_{c',s}) \pm O(\sigma_s \sqrt{n})) \quad (9.87) \\ &= \sqrt{\frac{2}{d}} \eta\zeta c \|\boldsymbol{v}_s\|^2 t \left(\frac{n_{c,s} - n_{c',s}}{n} \pm O(d^{-\Omega(\alpha)}) \right) \quad (9.88)\end{aligned}$$

The effect of the noise is captured by the $O(\sigma\sqrt{n})$ terms, following standard concentration inequalities, and we used the fact that $\frac{1}{\sqrt{n}} = O(d^{-\Omega(\alpha)})$. The result transfers to the full neural network under assumption 9.4.6, namely

$$f(\boldsymbol{v}_c;\boldsymbol{W}_t,\boldsymbol{z}_t) = \sqrt{\frac{2}{d}}\eta\zeta c\|\boldsymbol{v}_c\|^2 t\left(\frac{n_c}{n} \pm \mathcal{O}(d^{-\Omega(\alpha)})\right), \tag{9.89}$$

$$f(\boldsymbol{v}_s;\boldsymbol{W}_t,\boldsymbol{z}_t)=\sqrt{\frac{2}{d}}\eta\zeta c\|\boldsymbol{v}_s\|^2 t\left(\frac{n_{c,s}-n_{c',s}}{n} \pm \mathcal{O}(d^{-\Omega(\alpha)})\right), \tag{9.90}$$

This proves Theorem 5.3.1.

In addition, we calculate that

$$\beta_{norm}(t) = (tI+\boldsymbol{\Delta}_1)\sum_{i=1}^{n} y_i\left(\vartheta_0 + \vartheta_1(\frac{\|\boldsymbol{x}_i\|}{\sqrt{d}}-1)+\vartheta_2(\frac{\|\boldsymbol{x}_i\|}{\sqrt{d}}-1)^2\right) = O(\frac{\eta t}{\sqrt{n}})$$

Then for the predictions at time t for an example in class $c=1$, group $g_{1,s}$:

$$\begin{aligned}\boldsymbol{\psi}(\boldsymbol{x})^\top\boldsymbol{\beta}(t) &= \sqrt{\frac{2}{d}}\zeta\boldsymbol{x}^\top\boldsymbol{\beta'} + \sqrt{\frac{3}{2d}}\nu\beta_{bias}(t) + \beta_{norm}(t)\left(\vartheta_0+\vartheta_1(\frac{\|\boldsymbol{x}\|}{\sqrt{d}}-1)+\vartheta_2(\frac{\|\boldsymbol{x}\|}{\sqrt{d}}-1)^2\right)\\ &= \sqrt{\frac{2}{d}}\zeta(\boldsymbol{v}_1+\boldsymbol{v}_s+\boldsymbol{\xi})^\top\boldsymbol{\beta'} + \sqrt{\frac{3}{2d}}\nu\beta_{bias}(t) + \vartheta_0\beta_{norm}(t) \pm O\left(\eta t\sqrt{\frac{\log n}{nd}}\right)\end{aligned}$$

We have a few cases

1. $g_{1,k}$ is a majority group. In this case

$$\begin{aligned}\boldsymbol{\psi}(\boldsymbol{x})^\top\boldsymbol{\beta}(t) \geq \frac{2\eta\zeta^2 t}{d}&\left(\frac{n_1\|\boldsymbol{v}_c\|^2}{n} + \frac{\|\boldsymbol{v}_s\|^2(K-k)}{n} + \left\langle\boldsymbol{\xi},\frac{1}{n}\boldsymbol{X}^\top\boldsymbol{y}\right\rangle \pm O(d^{-\Omega(\alpha)})\right)\\ &+\sqrt{\frac{3}{2d}}\nu\beta_{bias}(t)+\vartheta_0\beta_{norm}(t)\pm O\left(\eta t\sqrt{\frac{\log n}{nd}}\right)\end{aligned}$$

2. $g_{1,k}$ is a minority group and $g_{-1,k}$ is a majority group. In this case

$$\begin{aligned}\boldsymbol{\psi}(\boldsymbol{x})^\top\boldsymbol{\beta}(t) \leq \frac{2\eta\zeta^2 t}{d}&\left(\frac{n_1\|\boldsymbol{v}_c\|^2}{n} - \frac{\|\boldsymbol{v}_s\|^2(K-k)}{n} + \left\langle\boldsymbol{\xi},\frac{1}{n}\boldsymbol{X}^\top\boldsymbol{y}\right\rangle \pm O(d^{-\Omega(\alpha)})\right)\\ &+\sqrt{\frac{3}{2d}}\nu\beta_{bias}(t)+\vartheta_0\beta_{norm}(t)\pm O\left(\eta t\sqrt{\frac{\log n}{nd}}\right)\end{aligned}$$

3. $g_{1,k}$ is such that no majority groups have the spurious feature. In this case

$$\boldsymbol{\psi}(\boldsymbol{x})^\top \boldsymbol{\beta}(t) = \frac{2\eta\zeta^2 t}{d}\left(\frac{n_1\|\boldsymbol{v}_c\|^2}{n} + \frac{\|\boldsymbol{v}_s\|^2\tilde{k}}{n} + \left\langle \boldsymbol{\xi}, \frac{1}{n}\boldsymbol{X}^\top \boldsymbol{y}\right\rangle \pm O(d^{-\Omega(\alpha)})\right)$$
$$+ \sqrt{\frac{3}{2d}}\nu\beta_{bias}(t) + \vartheta_0\beta_{norm}(t) \pm O\left(\eta t\sqrt{\frac{\log n}{nd}}\right), \qquad |\tilde{k}| \leq k$$

Now

$$\left\langle \boldsymbol{\xi}, \frac{1}{n}\boldsymbol{X}^\top \boldsymbol{y}\right\rangle = \sum_{c\in\{\pm 1\}} \frac{\|\boldsymbol{v}_c\| n_c}{n}\langle \boldsymbol{\xi}, \boldsymbol{v}_c\rangle + \sum_s \frac{\|\boldsymbol{v}_s\|(n_{1,s} - n_{-1,s})}{n}\langle \boldsymbol{\xi}, \boldsymbol{v}_s\rangle + \left\langle \boldsymbol{\xi}, \frac{1}{n}\sum_{i=1}^n \boldsymbol{\xi}_i y_i\right\rangle \tag{9.91}$$

$$= \sum_{c\in\{\pm 1\}} \frac{\|\boldsymbol{v}_c\| n_c}{n}\langle \boldsymbol{\xi}, \boldsymbol{v}_c\rangle + \sum_s \frac{\|\boldsymbol{v}_s\|(n_{1,s} - n_{-1,s})}{n}\langle \boldsymbol{\xi}, \boldsymbol{v}_s\rangle \pm O\left(\sqrt{\frac{d}{n}}\right) \tag{9.92}$$

$$\sim \mathcal{N}(0, \kappa) \pm O(d^{-\Omega(\alpha)}) \tag{9.93}$$

Finally, observe that $O\left(\eta t\sqrt{\frac{\log n}{nd}}\right) = O(d^{-1-\Omega(\alpha)})$. Combining all these results and setting $\rho_1 = \frac{2\eta\zeta^2 ct}{d}$, $\rho_2 = \frac{\rho_1 n_1\|\boldsymbol{v}_c\|^2}{n} + \sqrt{\frac{3}{2d}}\nu\beta_{bias}(t) + \vartheta_0\beta_{norm}(t)$ shows Theorem 5.3.2 when looking at the prediction of the linear model. Recall that [71] showed that the average squared error in predictions between the linear model and the full neural network is $O(\frac{\eta^2 t^2}{d^{2+\Omega(\alpha)}})$. Then by Markov's inequality, we can guarantee that the predictions of the linear model differ by at most $O(\frac{\eta t}{d^{1+\Omega(\alpha)}})$ for at least $1 - O(d^{-\Omega(\alpha)})$ proportion of the examples. This error can be factored into the existing error term. Hence the result holds for the full neural network.

We can apply the same argument for the class c'. Thus Theorem 5.3.2 is proven.

Notably, Theorem 5.3.2 only depends on the closeness of the neural network and the initial linear model on the training data, hence does not rely on assumption 9.4.6.

9.4.3.3 Proof of Theorem 5.3.3

Theorem 5.3.3. *Under the assumptions of Theorem 5.3.1, if the classes are balanced, and the total size of the minority groups in class c is small, i.e.,* $\mathcal{O}(n^{1-\gamma})$ *for some* $\gamma > 0$*, then there exists a constant* $\nu_2 > 0$ *such that at* $T = \nu_2 \cdot \frac{d \log d}{\eta}$*, for an example* $\boldsymbol{x}_i$ *in a majority group* $g_{c,s}$*, the contribution of the core feature to the model's output is at most:*

$$|f(\boldsymbol{v}_c; \boldsymbol{W}_T, \boldsymbol{z}_T)| \leq \sqrt{d}\frac{R_s}{\zeta R_c} + \mathcal{O}(n^{-\gamma}\sqrt{d}) + \mathcal{O}(d^{-\Omega(\alpha)}). \tag{5.9}$$

In particular if $\min\{R_c, 1\} \gg R_s$*, then the model's output is mostly indicated by the spurious feature instead of the core feature:*

$$|f(\boldsymbol{v}_s; \boldsymbol{W}_T, \boldsymbol{z}_T)| \geq \frac{\sqrt{d}}{2\zeta} \gg |f(\boldsymbol{v}_c; \boldsymbol{W}_T, \boldsymbol{z}_T)|. \tag{5.10}$$

Let g_{maj} be the total number of majority groups among all classes. Note that by the definition of majority groups, g_{maj} is at most the number of classes, namely 2 in the given analysis.

Since the classes are balanced with labels ±1, it is not hard to see that the bias term in the weights will always be zero, hence we may as well assume that we do not have the bias term. Abusing notation, we will still denote quantities by the same symbol, even though now the bias term has been removed.

First consider a model $\tilde{f} = \boldsymbol{\psi}^\top \tilde{\boldsymbol{\beta}}$ trained on the dataset $\mathcal{D}_{\text{maj}}$, which only contains examples from the majority groups. Further, assume $\mathcal{D}_{\text{maj}}$ has infinitely many examples so that the noise perfectly matches the underlying distribution. We prove the results in this simplified setting then extend the result using matrix perturbations.

We have

$$\mathcal{L} = \frac{1}{2}\mathbb{E}_{\mathcal{D}_{\text{maj}}}[(\boldsymbol{\psi}_i^\top \tilde{\boldsymbol{\beta}} - y_i)^2]$$
$$\nabla\mathcal{L} = \mathbb{E}_{\mathcal{D}_{\text{maj}}}[(\boldsymbol{\psi}_i^\top \tilde{\boldsymbol{\beta}} - y_i)\boldsymbol{\psi}_i]$$

and the optimal $\tilde{\boldsymbol{\beta}}_*$ satisfies

$$\tilde{\boldsymbol{\beta}}_* = \left(\mathbb{E}_{\mathcal{D}_{\text{maj}}}[\boldsymbol{\psi}_i \boldsymbol{\psi}_i^\top]\right)^\dagger \mathbb{E}_{\mathcal{D}_{\text{maj}}}[y_i \boldsymbol{\psi}_i]$$

where † represents the Moore-Penrose pseudo-inverse.

Since the noise is symmetrical with respect to the classes, the bias and norm terms of $\boldsymbol{\beta}$ must be zero. Thus the loss becomes

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\mathbb{E}_{(\boldsymbol{x}_i, y_i)\sim\mathcal{D}_{\text{maj}}}\left[(\sqrt{\frac{2}{d}}\zeta \boldsymbol{x}_i^\top \tilde{\boldsymbol{\beta}}' - y_i)^2\right] && (9.94)\\
&= \frac{1}{2}\mathbb{E}_{\mathcal{D}_{\text{maj}}}\left[(\sqrt{\frac{2}{d}}\zeta (\boldsymbol{v}_{c_i} + \boldsymbol{v}_{s_i} + \boldsymbol{\xi}_i)^\top \tilde{\boldsymbol{\beta}}' - y_i)^2\right] && (9.95)\\
&= \frac{1}{2}\mathbb{E}_{\mathcal{D}_{\text{maj}}}\left[(\sqrt{\frac{2}{d}}\zeta (\boldsymbol{v}_{c_i} + \boldsymbol{v}_{s_i})^\top \tilde{\boldsymbol{\beta}}' - y_i)^2 + (\sqrt{\frac{2}{d}}\zeta \boldsymbol{\xi}_i^\top \tilde{\boldsymbol{\beta}}')^2\right] && (9.96)\\
&= \frac{1}{2}\mathbb{E}_{\mathcal{D}_{\text{maj}}}\left[(\sqrt{\frac{2}{d}}\zeta (\boldsymbol{v}_{c_i} + \boldsymbol{v}_{s_i})^\top \tilde{\boldsymbol{\beta}}' - y_i)^2\right] + \frac{\zeta^2}{d}\tilde{\boldsymbol{\beta}}'^\top \boldsymbol{\Sigma}_\xi \tilde{\boldsymbol{\beta}}' && (9.97)
\end{aligned}$$

Consider the model $\boldsymbol{\beta}_s$ which only learns the spurious features of majority groups

$$\boldsymbol{\beta}'_s = \sqrt{\frac{d}{2}}\frac{1}{\zeta} \sum_{g_{c,s}\text{is a majority group}} \frac{c\boldsymbol{v}_s}{\|\boldsymbol{v}_s\|^2}.$$

Note that for any example in a majority group, $(\boldsymbol{v}_{c_i} + \boldsymbol{v}_{s_i})^\top \boldsymbol{\beta}'_s - y_i = 0$. Thus

$$\begin{aligned}
\mathcal{L} &= \frac{\zeta^2}{d}\tilde{\boldsymbol{\beta}}'^\top \boldsymbol{\Sigma}_\xi \tilde{\boldsymbol{\beta}}'\\
&= \sum_{\boldsymbol{v}_s\text{is spurious}} \frac{\sigma_s^2}{2\|\boldsymbol{v}_s\|^2}\\
&\leq \frac{g_{\text{maj}} R^2}{2}
\end{aligned}$$

The loss for the optimal model must be smaller. But the loss due to the last term in equation 9.97 along a core feature alone is

$$\frac{\zeta^2\sigma_c^2}{\|\boldsymbol{v}_c\|^2 d}\left\langle \boldsymbol{v}_c, \boldsymbol{\beta}'_*\right\rangle^2 \leq \frac{g_{\text{maj}}R^2}{2}$$

Rearranging gives

$$\left\langle \boldsymbol{v}_c, \boldsymbol{\beta}'_*\right\rangle^2 \leq \frac{d g_{\text{maj}} R^2\|\boldsymbol{v}_c\|^2}{2\zeta^2\sigma_c^2} \tag{9.98}$$

Now consider the loss from the first term in equation 9.97 due to a majority group. It must be at least

$$\begin{aligned}
\frac{K}{n}\left(1-\sqrt{\frac{2}{d}}\zeta\left\langle \boldsymbol{v}_s, \boldsymbol{\beta}'_*\right\rangle - \frac{\sqrt{g_{\text{maj}}}R\|\boldsymbol{v}_c\|}{\sigma_c}\right)^2 &\leq \frac{g_{\text{maj}}R^2}{2} \\
1-\sqrt{\frac{2}{d}}\zeta\left\langle \boldsymbol{v}_s, \boldsymbol{\beta}'_*\right\rangle - \frac{\sqrt{g_{\text{maj}}}R\|\boldsymbol{v}_c\|}{\sigma_c} &\leq \sqrt{\frac{n g_{\text{maj}}R^2}{2K}} \\
1-\sqrt{g_{\text{maj}}}R\left(\frac{\|\boldsymbol{v}_c\|}{\sigma_c}+\sqrt{\frac{n}{2K}}\right) &\leq \sqrt{\frac{2}{d}}\zeta\left\langle \boldsymbol{v}_s, \boldsymbol{\beta}'_*\right\rangle
\end{aligned}$$

Note that $\sqrt{\frac{n}{2K}} \leq \sqrt{\frac{g_{maj}}{2}}$. Now if we have R sufficiently smaller than $\frac{\sigma_c}{\sqrt{g_{maj}}\|\boldsymbol{v}_c\|}$ and $\frac{2}{g_{\text{maj}}}$, we can guarantee that the RHS is at least some constant less than 1, say $\frac{1}{\sqrt{2}}$. In this case, we have

$$\left\langle \boldsymbol{v}_s, \boldsymbol{\beta}_*\right\rangle^2 \geq \frac{d}{4\zeta^2} \tag{9.99}$$

Under these assumptions it is clear from equation 9.98 that we will also have

$$\frac{d}{4\zeta^2} \gg \left\langle \boldsymbol{v}_c, \boldsymbol{\beta}_*\right\rangle^2 \tag{9.100}$$

Now we return to the original dataset, which contains minority groups and only a finite number of examples. Again, we have

$$\boldsymbol{\beta}_* = (\boldsymbol{\Psi}^\top \boldsymbol{\Psi})^\dagger \boldsymbol{\Psi}^\top \boldsymbol{y}$$

Since we have removed the bias term, it is not hard to show that the matrix $\frac{1}{n}\boldsymbol{\Psi}^\top\boldsymbol{\Psi}$ has all eigenvalues of order $\Theta(\frac{1}{d})$. Now consider the difference between $\|\frac{1}{n}\boldsymbol{\Psi}^\top\boldsymbol{\Psi}\|$ and $\|\mathbb{E}_{\mathcal{D}_{\text{maj}}}[\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top]\|$. With high probability it will be of order $O(\frac{n_{\text{mino}}}{nd} + \frac{1}{d\sqrt{n}}) = O(\frac{n^{-\gamma}}{d})$, where the first term corresponds to the inclusion of minority groups and the second term corresponds having a finite sample size. It follows that

$$\begin{aligned}\left\|(\frac{1}{n}\boldsymbol{\Psi}^\top\boldsymbol{\Psi})^\dagger - (\mathbb{E}_{\mathcal{D}_{\text{maj}}}[\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top])^\dagger\right\| &= O\left(d - \frac{d}{d - O(n^{-\gamma})}\right) \\ &= O(dn^{-\gamma})\end{aligned}$$

A similar argument shows that

$$\|\boldsymbol{\Psi}^\top\boldsymbol{y} - \mathbb{E}_{\mathcal{D}_{\text{maj}}}[y_i\boldsymbol{\psi}_i]\| = O(d^{-\frac{1}{2}}n^{-\gamma})$$

Thus the change in alignment with a feature $\boldsymbol{v}$ is

$$\begin{aligned}\left\|\left\langle\tilde{\boldsymbol{\beta}}_*, \boldsymbol{v}\right\rangle - \langle\boldsymbol{\beta}_*, \boldsymbol{v}\rangle\right\| &= \left\|(\boldsymbol{\Psi}^\top\boldsymbol{\Psi})^\dagger\boldsymbol{\Psi}^\top\boldsymbol{y} - \left(\mathbb{E}_{\mathcal{D}_{\text{maj}}}[\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top]\right)^\dagger \mathbb{E}_{\mathcal{D}_{\text{maj}}}[y_i\boldsymbol{\psi}_i]\right\| \|\boldsymbol{v}\| \\ &\leq \left\| \left((\boldsymbol{\Psi}^\top\boldsymbol{\Psi})^\dagger - \left(\mathbb{E}_{\mathcal{D}_{\text{maj}}}[\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top]\right)^\dagger\right)\boldsymbol{\Psi}^\top\boldsymbol{y}\right. \\ &\qquad \left. + \left(\mathbb{E}_{\mathcal{D}_{\text{maj}}}[\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top]\right)^\dagger \left(\boldsymbol{\Psi}^\top\boldsymbol{y} - \mathbb{E}_{\mathcal{D}_{\text{maj}}}[y_i\boldsymbol{\psi}_i]\right)\right\| \|\boldsymbol{v}\| \\ &\leq O\left((dn^{-\gamma})(d^{-\frac{1}{2}}) + d(d^{-\frac{1}{2}}n^{-\gamma})\right) \\ &\leq O(n^{-\gamma}\sqrt{d})\end{aligned}$$

Replacing g_{maj} with 2,and combining equations 9.98, 9.99 9.101, and ASsumption 9.4.6, we get

$$|f(\boldsymbol{v}_s; \boldsymbol{W}_T, \boldsymbol{z}_T)| \geq \frac{\sqrt{d}}{2\zeta} \gg \sqrt{d}\frac{R_s}{\zeta R_c} + \mathcal{O}(n^{-\gamma}\sqrt{d}) + \mathcal{O}(d^{-\Omega(\alpha)}) \geq |f(\boldsymbol{v}_c; \boldsymbol{W}_T, \boldsymbol{z}_T)|. \qquad (9.101)$$

which proves the theorem.

9.4.4 Experimentation Details

9.4.4.1 Datasets

CMNIST We created a colored MNIST dataset with spurious correlations by using colors as spurious attributes following the settings in [229]. First, we defined an image classification task with 5 classes by grouping consecutive digits (0 and 1, 2 and 3, 4 and 5, 6 and 7, 8 and 9) into the same class. From the train split, we randomly selected 50,000 examples as the training set, while the remaining 10,000 samples were used as the validation set. The test split follows the official test split of MNIST.

For each class y_i, we assigned a color $\boldsymbol{v}_s$ from a set of colors $\mathcal{A}$={`#ff0000`, `#85ff00`, `#00fff3`, `#6e00ff`, `#ff0018`} as the spurious attribute that highly correlates with this class, represented by their hex codes, to the foreground of a fraction p_{corr} of the training examples. This fraction represents the majority group for class y_i. The stronger the spurious correlation between class y_i and the spurious attribute $\boldsymbol{v}_s$, the higher the value of p_{corr}. The remaining $1 - p_{corr}$ training examples were randomly colored using a color selected from $\mathcal{A} \setminus \boldsymbol{v}_s$. In our experiments, we set $p_{corr} = 0.995$ to establish significant spurious correlations within the dataset.

Waterbirds is introduced by [155] to study the spurious correlation between the background (land/water) and the foreground (landbird/waterbird) in image recognition. Species in Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset [195] are grouped into two classes, waterbirds and landbirds. All birds are then cut and pasted onto new background images, with waterbirds more likely to appear on water and landbirds having a higher probability on land. There are

4795 training examples in total, 3498 for landbirds with land background, 184 for landbirds with water background, 56 for waterbirds with land background, and 1057 for waterbirds with water background.

CelebA is a large-scale face attribute dataset comprised of photos of celebrities. Each image is annotated with 40 binary attributes, in which “blond hair" and “male" are commonly used for studying spurious correlations. Specifically, gender is considered a spurious feature for hair color classification. The smallest group is blond male.

9.4.4.2 Hyperparameters

The hyperparameters employed in our experiments on spurious benchmarks are detailed in Table 9.7. For the Waterbirds and CelebA datasets, we tuned the learning rate within the range of {1e-4, 1e-5} and weight decay within the range of {1e-1, 1e-0}. These ranges were determined based on the ranges of optimal hyperparameters used by the current state-of-the-art algorithms [43, 106, 155, 133, 229]. The batch sizes and total training epochs remained consistent with those used in these prior studies. To determine the epoch for separating groups, we performed clustering on the validation set while training the model on the training set to maximize the minimum recall of SPARE's clusters with the groups in the validation set. As mentioned in Section 5.5.2, we decided the number of clusters and adjusted the sampling power for each class based on Silhouette scores. Specifically, when the Silhouette score was below 0.9, a sampling power of 2 was applied, while a sampling power of 1 was used otherwise. It is important to note that other algorithms tuned hyperparameters, such as epochs to separate groups and upweighting factors, by maximizing the worst-group accuracy of fully trained models on the validation set, which is more computationally demanding than the hyperparameter tuning of SPARE.

Table 9.7: Hyperparameters used for the reported results on different datasets.

Dataset	CMNIST	Waterbirds	CelebA
Learning rate	1e-3	1e-4	1e-5
Weight decay	1e-3	1e-1	1e-0
Batch size	32	128	128
Training epochs	20	300	50
Group separation epoch	2	2	1
Silhouette scores	[0.997,0.978,0.996,0.991,0.996]	[0.886,0.758]	[0.924,0.757]
Sampling power	[1,1,1,1,1]	[2,2]	[1,2]

9.4.4.3 Choices of Model Outputs

In our experiments, we found the worst-group accuracy gets the most improvement when Spare uses the outputs of the last linear layer to separate the majority from the minority for CMNIST and Waterbirds and use the second to last layer (i.e., the feature embeddings inputted to the last linear layer) to identify groups in CelebA. We speculate that this phenomenon can be attributed to the increased complexity of the CelebA dataset compared to the other two datasets, as employing a higher output dimension help identify groups more effectively.

9.4.5 Discovering Spurious Features

9.4.5.1 Restricted ImageNet

We use Restricted ImageNet proposed in [187] which contains 9 superclasses of ImageNet. The classes and the corresponding ImageNet class ranges are shown in Table 9.8.

9.4.5.2 Experimental Settings

When training on Restricted ImageNet, we use ResNet50 [67] from the PyTorch library [141] with randomly initialized weights instead of pretrained weights. We followed the hyperparameters specified in [63]: the model was trained for 90 epochs, with an initial

Table 9.8: Classes included in Restricted ImageNet and their corresponding ImageNet class ranges.

Restricted ImageNet Class	ImageNet class range
dog	151-268
cat	281-285
frog	30-32
turtle	33-37
bird	80-100
primate	365-382
fish	389-397
crab	118-121
insect	300-319

learning rate of 0.1. The learning rate was reduced by a factor of 0.1 at the 30th, 60th, and 80th epochs. During training, we employed Nesterov momentum of 0.9 and applied a weight decay of 0.0001.

9.4.5.3 Investigation on Groups Identified by EIIL vs. Spare

Evaluation setup. As no group-labeled validation set is available to tune the epoch in which the groups are separated, we tried separating groups using ERM models trained for various numbers of epochs. Since both EIIL and Spare identify the groups early (EIIL infers groups on models trained with ERM for 1 epoch for both Waterbirds and CelebA, as shown in Table 9.12 and Table 9.11, and 5 epochs for CMNIST; the group separation epochs for Spare are epoch 1 or 2 for the three datasets, as shown in Table 9.7), we tuned the epoch to separate groups in the range of {2,4,6,8} for both algorithms. This tuning was based on the average test accuracy achieved by the final model, as the worst-group accuracy is undefined without group labels. Interestingly, while Spare did not show sensitivity to the initial epochs on Restricted ImageNet, EIIL achieved the highest average test accuracy when the initial models were trained for 4 epochs using ERM.

EIIL finds groups of misclassified examples while Spare finds groups with spurious features. We observed that EIIL effectively separates examples that have 0%

Table 9.9: Accuracy (%) of training examples in different classes of Restricted ImageNet in the two environments inferred by EIIL. EIIL trains models with Group DRO on the inferred environments, resulting in up-weighting misclassified examples in Env 2.

Class	dog	cat	frog	turtle	bird	primate	fish	crab	insect
Env 1 ERM acc	98	37	26	62	76	78	78	71	90
Env 2 ERM acc	0	0	0	0	0	0	0	0	0
Env 1 size	144378	488	457	2875	17157	11233	6817	2172	21112
Env 2 size	3495	6012	3443	3625	9984	12167	4417	3028	4888

classification accuracy as the minority group, as demonstrated in Table 9.9. This separation is analogous to the error-splitting strategy employed by JTT [106] when applied to the same initial model. This similarity in behavior is also discussed in [43]. Instead of focusing on misclassified examples, SPARE separates the examples that are learned early in training. Table 9.10 shows that the first cluster found by SPARE have almost 100% accuracy, indicating that the spurious feature is learned for such examples. Downweighting examples that are learned early allows for effectively mitigating the spurious correlation.

SPARE upweights outliers less than EIIL. Heavily upweighting misclassified examples can be problematic for this more realistic dataset than the spurious benchmarks as the misclassified ones are likely to be outliers, noisy-labeled or contain non-generalizable information. Table 9.9 shows that groups inferred by EIIL are more imbalanced, which makes EIIL upweights misclassified examples more than SPARE. As shown in Table 5.4, this heavier upweighting of misclassified examples with EIIL drops accuracy not only for the minority groups but also for the overall accuracy. Therefore, we anticipate that this effect would persist or become even more pronounced for methods like JTT, which directly identify misclassified examples as the minority group. In contrast, SPARE separates groups based on the spurious feature that is learned early, and upweights the misclassified examples less than other methods due to the more balanced size of the clusters. This allows SPARE to more effectively mitigate spurious correlations than others.

Table 9.10: Accuracy (%) of training examples in different classes of Restricted ImageNet in the two groups inferred by Spare at epoch 8.

Class	dog	cat	frog	turtle	bird	primate	fish	crab	insect
Cluster 1 ERM acc	100	100	100	100	100	99	100	100	100
Cluster 2 ERM acc	64	9	11	14	28	13	27	16	36
Cluster 1 size	130541	3236	1578	2684	18870	12158	7331	2566	18974
Cluster 2 size	17332	3264	2322	3816	8271	11242	3903	2634	7026

9.4.6 Comparing Inferred with Ground-truth Groups

In Table 9.11 and Table 9.12, we compare the clusters found by Spare vs. (1) misclassified examples found by JTT, (2) environments inferred by EIIL, and (3) pseudo-labels learned by SSA.

9.4.6.1 Implementation of Baselines

Both JTT [106] and EIIL [43] require training an ERM model to identify groups of examples for upweighting or downweighting. For clarity, we will refer to this ERM model as the *reference model*, which is equivalent to the *identification models* defined in [106].

JTT. We train the reference model from ImageNet-pre-trained weights with ERM based on the optimal hyperparameters reported in [106] and upsample training examples misclassified by the identification models. For Waterbirds, we train the identification model for 60 epochs with a learning rate 1e-5 and weight decay 1. For CelebA, the identification model is trained for 1 epoch with a learning rate 1e-5 and weight decay 0.1.

EIIL. For Waterbirds, we follow the environment inference steps explained in [43]: we use an ERM model trained for 1 epoch as the reference model and optimize the EI objective of EIIL with learning rate 0.01 for 20, 000 steps using the Adam optimizer. As no experiment was conducted on CelebA in the original paper [43], we follow the proposal in [133], which took the same EI procedure for CelebA as for Waterbirds.

SSA. We implement SSA based on the pseudo-code and experimental details explained in [133]. Please refer to [133] for details of the setups. As the pseudo-attribute predictor shares the same architecture as the robust model but is trained on the validation set, to make the inference cost comparable across all methods, we report the inference cost of SSA by converting the number of training-on-validation steps for the pseudo-attribute predictor to the number of training-on-train epochs that involve the same total number of gradient backward steps.

9.4.6.2 Comparison of Groups.

CelebA. We start from the CelebA dataset, where we observed more significant disparities among the groups identified by different algorithms, as demonstrated in Table 9.11. JTT simply upweights the smaller class (i.e., blond hair), as most examples from that class are misclassified due to the strong class imbalance. Similarly, EIIL assigns higher weights to more examples from the smaller class.

On the other hand, when examining the confusion matrices, we found that both SSA and Spare successfully discover groups that closely align with the ground-truth groups in CelebA. Note that Spare requires much less training than SSA. However, upon visualizing the samples, we noticed that the upweighted examples identified by SSA exhibit some characteristics learned from the validation set that are more correlated with a certain gender. For instance, 11.4% of the upweighted examples of blonde females and only 1.2% of the downweighted examples wear sunglasses, which is a feature that is correlated more with males in the validation set (13.5% of males vs. only 2.3% of females in the validation set wear sunglasses). Importantly, when examining the correlation between hair colors (actual class labels) and sunglasses, we observe a milder correlation between non-blond hair and sunglasses: 7.3% of non-blond haired wear sunglasses compared to only 1.7% of those with blond hair. Therefore, the pseudo-attribute predictor has likely learned to correlate blond males with sunglasses, resulting in the potential to amplify other (potentially spurious) correlations learned from the validation set while mitigating the targeted spurious correlations.

Waterbirds. In line with our observations on CelebA, as shown in Table 9.12, the groups identified by JTT are similar to those identified by EIIL, and the groups identified by SSA share similarities with the groups identified by Spare, which requires less training. Specifically, JTT and EIIL focus on upweighting noisy and outlier examples, SSA upweights examples that may possess certain (spurious) features (i.e., yellow feathers), and Spare prioritizes upweighting minority groups that do not share the spurious features with the majority groups.

9.4.7 Reproducibility

Each experiment was conducted on one of the following GPUs: NVIDIA A40 with 45G memory, NVIDIA RTX A6000 with 48G memory, and NVIDIA RTX A5000 with 24G memory.

9.5 Appendix for Chapter 6

9.5.1 Synthetic Experiments

Datasets. We generate $10,000$ training examples and $10,000$ test examples from the data distribution defined in Definition 6.1.1 with dimension $d = 50$ and number of patches $P = 3$. Specifically, we let $\alpha = 0.98$, $\beta_c = 0.2$, $\beta_s = 1$ and $\sigma_p = 0.78$ for Table 6.1 as well as Figure 6.3a and Figure 6.3c. For Figure 6.3b, we consider a data distribution where $\alpha = 0.98$, $\beta_c = 1$, $\beta_s = 0.2$ and $\sigma_p = 0.78$. Furthermore, we randomly shuffle the order of the patches of $\mathbf{x}$ after we generate data $(\mathbf{x}, y, a)$.

Training. We consider the performances of a nonlinear CNN trained with ERM and PDE. The nonlinear CNN architecture follows (6.3) with the cubic activation function, where we let the number of neurons/filters $J = 40$. We use gradient descent with momentum (GD+M) as the optimizer of our method, setting the momentum to 0.9 and the learning rate to 0.03. The number of warm-up iterations is set to 800. We consider ERM trained with GD with a learning rate 0.1 and without momentum to align with our theoretical finding in both

Table 9.11: Comparison of groups found by different methods for CelebA.

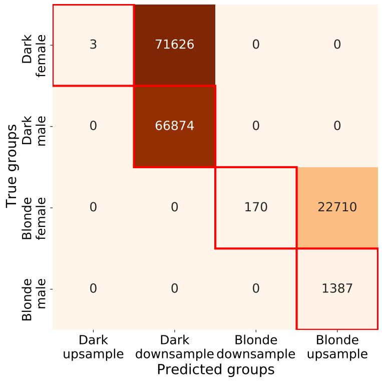
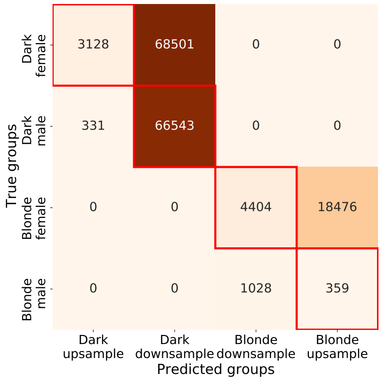
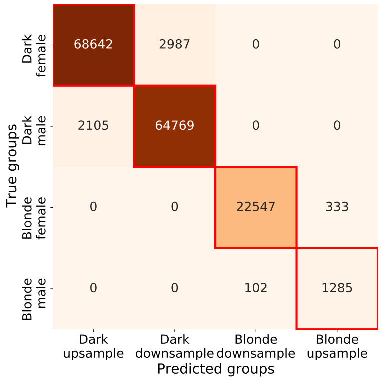
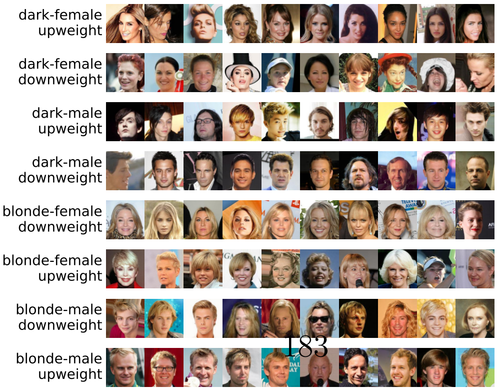
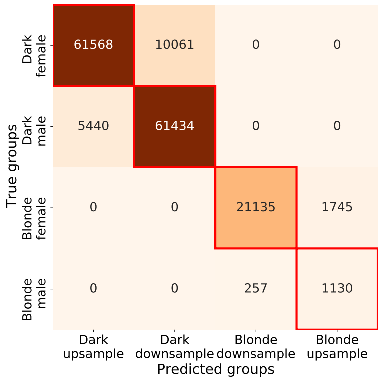
Inference (Cost)	Method	Samples	Confusion Matrix
JTT (1 epoch)		dark-female upweight dark-female downweight dark-male upweight dark-male downweight blonde-female downweight blonde-female upweight blonde-male downweight blonde-male upweight	
EIIL (1 epoch)		dark-female upweight dark-female downweight dark-male upweight dark-male downweight blonde-female downweight blonde-female upweight blonde-male downweight blonde-male upweight	
SSA (53 epochs)		dark-female upweight dark-female downweight dark-male upweight dark-male downweight blonde-female downweight blonde-female upweight blonde-male downweight blonde-male upweight	
SPARE (1 epoch)		 dark-female upweight dark-female downweight dark-male upweight dark-male downweight blonde-female downweight blonde-female upweight blonde-male downweight blonde-male upweight	

Table 9.12: Comparison of groups found by different methods for Waterbirds.

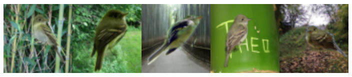
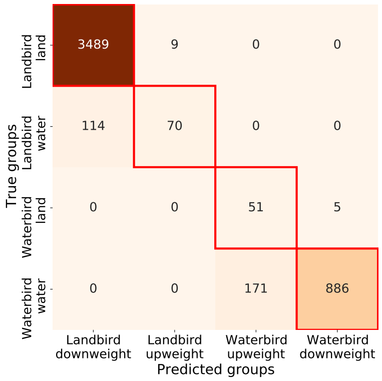
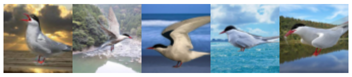
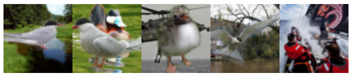
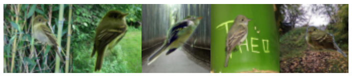
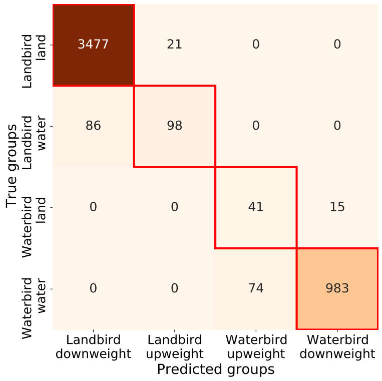
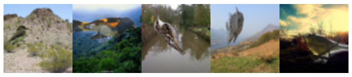
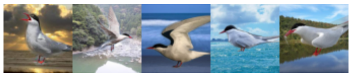
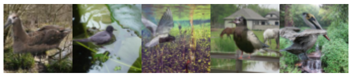
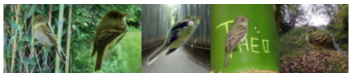
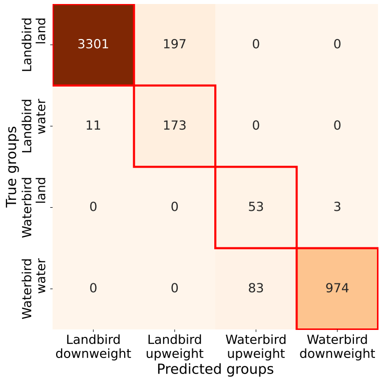
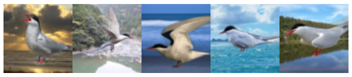
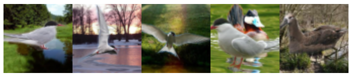
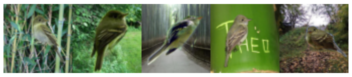
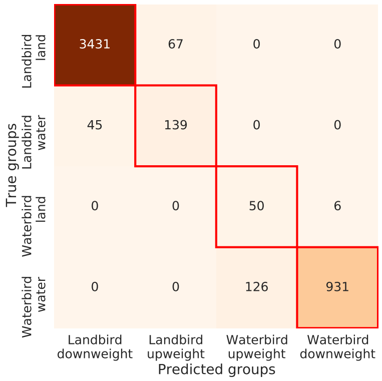
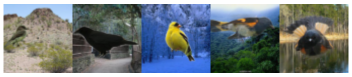
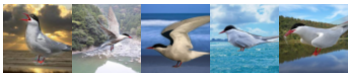
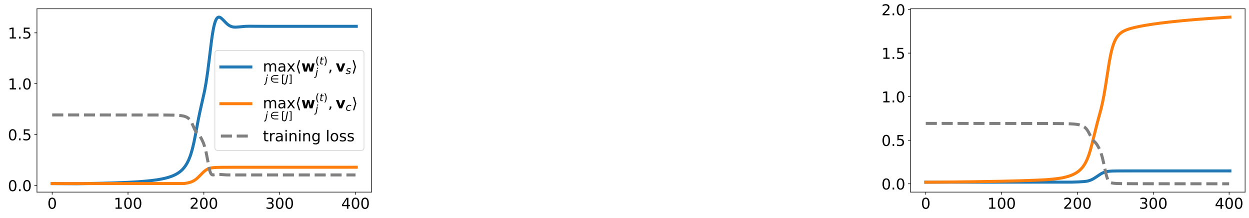
Inference (Cost)	Method	Samples	Confusion Matrix
JTT (60 epochs)	landbird-land downweight		
	landbird-land upweight		
	waterbird-water downweight		
	waterbird-water upweight		
EIIL (1 epoch)	landbird-land downweight		
	landbird-land upweight		
	waterbird-water downweight		
	waterbird-water upweight		
SSA (40 epochs)	landbird-land downweight		
	landbird-land upweight		
	waterbird-water downweight		
	waterbird-water upweight		
SPARE (1 epoch)	landbird-land downweight		
	landbird-land upweight		
	waterbird-water downweight		
	waterbird-water upweight		

Table 6.1 and Figure 6.4. In Table 6.1, we also show the experiment results for ERM trained with GD+M as same as PDE. All models are trained until convergence.

Additional experiments. In Figure 9.10, we demonstrate the growth of $\max_{j\in[J]}\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle$ and $\max_{j\in[J]}\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle$ for ERM trained with GD+M under the same data generated in Figure 6.4. Similarly, we observe that ERM learns the spurious feature quickly as the training loss is minimized under our data distribution. Meanwhile, if the data is generated as in case 2 where $\beta_c > \beta_s$, ERM learns the core feature correctly.



(a) ERM (case 1)

(b) ERM (case 2)

Figure 9.10: **Training process of ERM trained with GD+M.** We consider the same dataset generated in Figure 6.4 and observe almost the same training process as ERM with GD, except GD+M learns the features faster.

Furthermore, we consider the following variation of our methods on the same dataset in Table 6.1 to demonstrate the importance of gradual expansion. In Figure 9.11, we let PDE incorporate all of the new training data at once after the warm-up stage. As demonstrated, adding all data at once makes it harder for the model to continue learning core features, resulting in a worst-group accuracy of 74.24% as compared to 94.32% for progressive expansion.

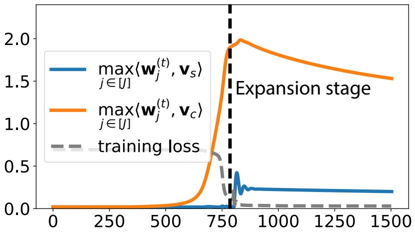


Figure 9.11: **Variation of PDE.** We consider the same dataset generated in Figure 6.4 and add all data at once after the warm-up stage.

9.5.2 Benchmark Datasets

Waterbirds. The Waterbirds dataset [155] was constructed to study object recognition models relying on image backgrounds instead of the object itself. To this end, bird images from the Caltech-UCSD Birds-200-2011 (CUB) dataset [196] were combined with backgrounds from the Places dataset [234]. The dataset contains $4,795$ bird images labeled as a waterbird or landbird and placed against a water or land background. Waterbirds are predominantly located against a water background, while landbirds are situated against a land background. Notably, the smallest subgroup in the dataset is waterbirds on land, consisting of only 56 examples.

CelebA. The CelebA dataset [109] is a popular face attribute dataset used to examine the spurious associations between non-demographic and demographic attributes. Specifically, one of the 40 binary attributes, “blond hair", is used as the target attribute, and “male" is the spurious attribute. The dataset contains $162,770$ training examples, with the smallest group being blond-haired males, with only 1387 examples.

CivilComments-WILDS. The CivilComments-WILDS dataset [93] is designed to explore the challenge of classifying online comments as either toxic or non-toxic while dealing with the spurious correlation between the label and demographic information such as gender, race, religion, and sexual orientation. The dataset's evaluation metric, as defined by [93], creates 16 overlapping groups for each of the eight demographic identities, resulting in a total of 512 distinct groups. For each group, the metric calculates the worst-case performance of a classifier, which allows for a robust evaluation of the model's ability to generalize across diverse populations.

9.5.3 Real Data Experiments

Setup. Our experiment settings strictly follow the same setting used for datasets introduced in Section 9.5.2 in previous works [155, 106, 132, 43, 90]. Specifically, we built our training pipeline with the WILDS package [92] which uses pretrained ResNet-50 model [67] in Pytorch

Table 9.13: Number of data in our warm-up dataset for PDE's results in Table 6.2. We also report the number of data in total for the three datasets.

Dataset	Warm-up	All
Waterbirds	224	4,795
CelebA	5,548	162,770
CivilComments-WILDS	13,705	269,038

[141] library for the image datasets (i.e., Waterbirds and CelebA) and Transformer [193] in Transformers library [202] for CivilComments-WILDS. All experiments were conducted on a single NVIDIA RTX A6000 GPU with 48GB memory.

Training. In Table 9.13, we summarize the number of data used in the warm-up stage for PDE in Table 6.2 with the total number of data in the entire datasets. In Table 9.14, we report the hyperparameters used for PDE with the notations in Algorithm 5. Specifically, T_0 refers to the number of epochs for the warm-up stage and J refers to the number of epochs for training after each data expansion. Lastly, m is the number of added data for each data expansion. Our batch size is consistent with GroupDRO.

Table 9.14: Hyperparameters used for PDE's results in Table 6.2. Note that T_0 and J are in epochs of PDE's training set, which have fewer iterations than epochs of the full training set.

Dataset	Learning rate	Weight decay	Batch size	T_0	J	m
Waterbirds	1e-2	1e-2	64	140	10	10
CelebA	1e-2	1e-4	128	16	10	50
CivilComments-WILDS	1e-5	1e-2	16	15	2	300

Groups for CivilComments-WILDS. We note that the demographic tags in CivilComments-WILDS can coexist in the input text. For example, a text can contain both tags of female and male. Therefore, combining the 8 demographic tags with the binary classification label (toxic vs. non-toxic) results in 16 overlapping groups, where each group counts as data from a class with/without a specific tag. For computational efficiency, previous methods divide the data into four non-overlapping groups either by the *specific* one demographic tag a_i (groups are $\{y = \pm 1, a_i = \pm 1\}$) [93] or by containing *any* one of the tags: $a = 1$ if any $a_i = 1$ and $a = -1$ otherwise (groups are $\{y = \pm 1, a = \pm 1\}$) [106, 43]. However, the data can actually be partitioned into 512 distinct groups, with each group corresponding to different combinations of tags: $\{y = \pm 1, a_1 = \pm 1, a_2 = \pm 1, \ldots, a_n = \pm 1\}$. As GroupDRO requires computation per group at each training batch, considering a large number of groups makes it harder for GroupDRO to train efficiently. Meanwhile, having more groups does not impose an additional computational cost on PDE, so we can consider all these data groups when constructing our warm-up set. As many groups are empty or contain very little data, we set a threshold to select at most 150 data points from each group to ensure a balanced yet sufficient warm-up set.

Efficiency. In Table 9.15, we further report the training efficiency of PDE compared with GroupDRO on CelebA and CivilComments-WILDS. Similar to what we observe on the Waterbirds dataset, PDE achieves the best performance at a larger learning rate and smaller weight decay on CelebA with a significant speedup as compared to GroupDRO. On CivilComments-WILDS, we can also observe an improved efficiency.

Table 9.15: Training efficiency of PDE and GroupDRO on CelebA dataset.

Method	Learning rate	Weight decay	Worst	Average	Early-stopping epoch*
GroupDRO	1e-5	1e-1	$86.3_{\pm 1.1}$	$92.9_{\pm 0.3}$	$23.7_{\pm 6.8}$
PDE	1e-2	1e-4	$\mathbf{91.0}_{\pm 0.4}$	$92.0_{\pm 0.6}$	$\mathbf{0.7}_{\pm 0.3}$

Table 9.16: Training efficiency of PDE and GroupDRO on CivilComments-WILDS dataset.

Method	Learning rate	Weight decay	Worst	Average	Early-stopping epoch*
GroupDRO	1e-5	1e-2	$69.4_{\pm 0.9}$	$89.6_{\pm 0.5}$	$3.3_{\pm 2.1}$
PDE	1e-5	1e-2	$\mathbf{71.5}_{\pm 0.5}$	$86.3_{\pm 1.7}$	$\mathbf{2.1}_{\pm 1.1}$

Data Augmentation. Additionally, the increased training speed of our method facilitates the usage of techniques such as data augmentation. While data augmentation is a common practice for improving model generalization, DRO approaches have not incorporated it into their methods. We hypothesize that this omission stems from the slower training process. Data augmentation introduces random noise to the training data, which complicates convergence during training when using a very small learning rate. As illustrated in Table 9.17, data augmentation leads to slightly worse performance for GroupDRO. In contrast, our method effectively benefits from data augmentation.

Table 9.17: The effect of data augmentation on GroupDRO and PDE on Waterbirds dataset. We report the worst-group and average accuracy.

	GroupDRO		PDE	
Method	Worst	Avg	Worst	Avg
W/o data aug	86.7	93.2	88.9	89.5
W/ data aug	85.7	96.6	**90.3**	**92.4**

9.5.4 Proof Preliminaries

Notation. In this paper, we use lowercase letters, lowercase boldface letters, and uppercase boldface letters to respectively denote scalars (a), vectors ($\boldsymbol{v}$), and matrices ($\mathbf{W}$). We use sgn to denote the sign function.For a vector $\boldsymbol{v}$, we use $\|\boldsymbol{v}\|_2$ to denote its Euclidean norm. Given two sequences $\{x_n\}$ and $\{y_n\}$, we denote $x_n = \mathcal{O}(y_n)$ if $|x_n| \leq C_1|y_n|$ for some absolute positive constant C_1, $x_n = \Omega(y_n)$ if $|x_n| \geq C_2|y_n|$ for some absolute positive constant C_2, and $x_n = \Theta(y_n)$ if $C_3|y_n| \leq |x_n| \leq C_4|y_n|$ for some absolute constants $C_3, C_4 > 0$. We use $\tilde{\mathcal{O}}(\cdot)$ to hide logarithmic factors of d in $\mathcal{O}(\cdot)$.

Before we go into the analysis, we first consider the following gradient,

$$\nabla_{\mathbf{w}_j}\mathcal{L}(\mathbf{W}^{(t)}) = -\frac{1}{N}\sum_{i=1}^{N}\frac{\exp(-y_i f(\mathbf{x}_i;\mathbf{W}^{(t)}))}{1+\exp(-y_i f(\mathbf{x}_i;\mathbf{W}^{(t)}))}\cdot y_i f'(\mathbf{x}_i;\mathbf{W}^{(t)}). \tag{9.102}$$

Let's denote the derivative of a data example i at iteration t to be

$$\ell_i^{(t)} = \frac{\exp(-y_i f(\mathbf{x}_i;\mathbf{W}^{(t)}))}{1+\exp(-y_i f(\mathbf{x}_i;\mathbf{W}^{(t)}))} = \mathrm{sigmoid}(-y_i f(\mathbf{x}_i;\mathbf{W}^{(t)})). \tag{9.103}$$

Lemma 9.5.1. *(Gradient) Let the loss function $\mathcal{L}$ be as defined in (6.1). For $t \geq 0$ and*

$j \in [J]$, the gradient of the loss $\mathcal{L}$ with regard to neuron $\mathbf{w}_j$ is

$$\nabla_{\mathbf{w}_j}\mathcal{L}(\mathbf{W}^{(t)}) = -\frac{3}{N}\bigg(\beta_c^3\sum_{i=1}^{N}\ell_i^{(t)}\langle\mathbf{w}_j,\boldsymbol{v}_c\rangle^2\boldsymbol{v}_c + \sum_{i=1}^{N}\ell_i^{(t)}y_i\langle\mathbf{w}_j,\boldsymbol{\xi}_i\rangle^2\boldsymbol{\xi}_i + \Big(\sum_{i\in S_1}\ell_i^{(t)} - \sum_{i\in S_2}\ell_i^{(t)}\Big)\cdot\beta_s^3\langle\mathbf{w}_j,\boldsymbol{v}_s\rangle^2\boldsymbol{v}_s\bigg).$$

Proof. We have the following gradient

$$\nabla_{\mathbf{w}_j}\mathcal{L}(\mathbf{W}^{(t)}) = -\frac{1}{N}\sum_{i=1}^{N}\frac{\exp(-y_i f(\mathbf{x}_i;\mathbf{W}^{(t)}))}{1+\exp(-y_i f(\mathbf{x}_i;\mathbf{W}^{(t)}))}\cdot y_i f'(\mathbf{x}_i;\mathbf{W}^{(t)}). \tag{9.104}$$

And let's denote the derivative of a data example i at iteration t to be

$$\ell_i^{(t)} = \frac{\exp(-y_i f(\mathbf{x}_i;\mathbf{W}^{(t)}))}{1+\exp(-y_i f(\mathbf{x}_i;\mathbf{W}^{(t)}))} = \mathrm{sigmoid}(-y_i f(\mathbf{x}_i;\mathbf{W}^{(t)})). \tag{9.105}$$

Then, we can further write the gradient as

$$\begin{aligned}
\nabla_{\mathbf{w}_j}\mathcal{L}(\mathbf{W}^{(t)}) &= -\frac{3}{N}\sum_{i=1}^{N}\ell_i^{(t)}y_i\sum_{p=1}^{P}\langle\mathbf{w}_j,\mathbf{x}^{(p)}\rangle^2\cdot\mathbf{x}^{(p)} \\
&= -\frac{3}{N}\sum_{i=1}^{N}\ell_i^{(t)}y_i\Big(\langle\mathbf{w}_j,\beta_c y_i\boldsymbol{v}_c\rangle^2\beta_c y_i\boldsymbol{v}_c + \langle\mathbf{w}_j,\beta_s a_i\boldsymbol{v}_s\rangle^2\beta_s a_i\boldsymbol{v}_s + \langle\mathbf{w}_j,\boldsymbol{\xi}_i\rangle^2\boldsymbol{\xi}_i\Big) \\
&= -\frac{3}{N}\sum_{i=1}^{N}\ell_i^{(t)}\Big(\beta_c^3\langle\mathbf{w}_j,\boldsymbol{v}_c\rangle^2\boldsymbol{v}_c + \beta_s^3 y_i a_i\langle\mathbf{w}_j,\boldsymbol{v}_s\rangle^2\boldsymbol{v}_s + y_i\langle\mathbf{w}_j,\boldsymbol{\xi}_i\rangle^2\boldsymbol{\xi}_i\Big) \\
&= -\frac{3}{N}\Bigg(\sum_{i=1}^{N}\ell_i^{(t)}\Big(\beta_c^3\langle\mathbf{w}_j,\boldsymbol{v}_c\rangle^2\boldsymbol{v}_c + y_i\langle\mathbf{w}_j,\boldsymbol{\xi}_i\rangle^2\boldsymbol{\xi}_i\Big) \\
&\qquad + \Big(\sum_{i\in S_1}\ell_i^{(t)} - \sum_{i\in S_2}\ell_i^{(t)}\Big)\beta_s^3\langle\mathbf{w}_j,\boldsymbol{v}_s\rangle^2\boldsymbol{v}_s\Bigg),
\end{aligned}$$

where the last equality holds due to that for $i \in S_1$ we have $a_i = y_i$ and for $i \in S_2$ we have $a_i = -y_i$. □

With the gradient, we have the following:

Core feature gradient. The projection of the gradient on $\boldsymbol{v}_c$ is then

$$\langle \nabla_{\mathbf{w}_j} \mathcal{L}(\mathbf{W}^{(t)}), \boldsymbol{v}_c \rangle = -\frac{3\beta_c^3}{N} \sum_{i=1}^{N} \ell_i^{(t)} \langle \mathbf{w}_j, \boldsymbol{v}_c \rangle^2. \tag{9.106}$$

Spurious feature gradient. The projection of the gradient on $\boldsymbol{v}_s$ is

$$\langle \nabla_{\mathbf{w}_j} \mathcal{L}(\mathbf{W}^{(t)}), \boldsymbol{v}_s \rangle = -\frac{3\beta_s^3}{N} \Big(\sum_{i \in S_1} \ell_i^{(t)} - \sum_{i \in S_2} \ell_i^{(t)} \Big) \cdot \langle \mathbf{w}_j, \boldsymbol{v}_s \rangle^2. \tag{9.107}$$

Noise gradient. The projection of the gradient on $\boldsymbol{\xi}_i$ is

$$\langle \nabla_{\mathbf{w}_j} \mathcal{L}(\mathbf{W}^{(t)}), \boldsymbol{\xi}_i \rangle = -\frac{3y_i}{N} \sum_{i=1}^{N} \ell_i^{(t)} \langle \mathbf{w}_j, \boldsymbol{\xi}_i \rangle^2 \|\boldsymbol{\xi}_i\|_2^2. \tag{9.108}$$

Derivative of data example i. $\ell_i^{(t)}$ can be rewritten as

$$\begin{aligned} \ell_i^{(t)} &= \mathrm{sigmoid}\big(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)}) \big) \\ &= \mathrm{sigmoid}\Big(\sum_{j=1}^{J} -\beta_c^3 \langle \mathbf{w}_j, \boldsymbol{v}_c \rangle^3 - y_i a_i \beta_s^3 \langle \mathbf{w}_j, \boldsymbol{v}_s \rangle^3 - y_i \langle \mathbf{w}_j, \boldsymbol{\xi}_i \rangle^3 \Big). \end{aligned} \tag{9.109}$$

Note that $0 < \ell_i^{(t)} < 1$ due to the property of the sigmoid function. Furthermore, we similarly consider that the sum of the sigmoid terms for all time steps is bounded up to a logarithmic dependence [37]. The sigmoid term is considered small for a κ such that

$$\sum_{t=0}^{T} \frac{1}{1 + \exp(\kappa)} \leq \tilde{O}(1),$$

which implies $\kappa \geq \tilde{\Omega}(1)$.

9.5.5 Proof of Theorem 6.1.2

In this section, we present the detailed proofs that build up to Theorem 6.1.2. We begin by considering the update for the spurious feature and core feature.

Lemma 9.5.2 (Spurious feature update.)**.** *For all $t \geq 0$ and $j \in [J]$, the spurious feature update is*

$$\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s \rangle = \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle + \frac{3\eta\beta_s^3}{N}\Big(\sum_{i \in S_1} \ell_i^{(t)} - \sum_{i \in S_2} \ell_i^{(t)}\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2,$$

which gives

$$\begin{aligned}\tilde{\Theta}(\eta)\beta_s^3\Big(\hat{\alpha}g_1(t) - \sum_{i \in S_2} \ell_i^{(t)}/N\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2 \leq \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s \rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle \\ \leq \tilde{\Theta}(\eta)\beta_s^3 \cdot \hat{\alpha}g_1(t) \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2,\end{aligned}$$

where $g_1(t) = sigmoid\big(-\sum_{j \in [J]}(\beta_c^3\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle^3 + \beta_s^3 \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^3)\big)$.

Proof. The spurious feature update is obtained by using the gradient update of $\mathbf{W}^{(t)}$ and plugging in (9.107):

$$\begin{aligned}\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s \rangle &= \langle \mathbf{w}_j^{(t)} - \eta \nabla_{\mathbf{w}_j} \mathcal{L}(\mathbf{W}^{(t)}), \boldsymbol{v}_s \rangle \\ &= \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle + \frac{3\eta\beta_s^3}{N}\Big(\sum_{i \in S_1} \ell_i^{(t)} - \sum_{i \in S_2} \ell_i^{(t)}\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2.\end{aligned}$$

We first prove the upper bound. Consider the following,

$$\begin{aligned}\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s \rangle &= \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle + \frac{3\eta\beta_s^3}{N}\Big(\sum_{i \in S_1} \ell_i^{(t)} - \sum_{i \in S_2} \ell_i^{(t)}\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2 \\ &\leq \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle + \frac{3\eta\beta_s^3}{N}\Big(\sum_{i \in S_1} \ell_i^{(t)}\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2 \\ &\leq \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle + \tilde{\Theta}(\eta)\beta_s^3 \cdot \frac{\sum_{i \in S_1} g_1(t)}{N} \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2 \\ &= \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle + \tilde{\Theta}(\eta)\beta_s^3\hat{\alpha} \cdot g_1(t) \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2,\end{aligned}$$

where the first inequality holds due to $0 < \ell_i^{(t)} < 1$, the second inequality holds due to Lemma 9.5.12, and the last equality holds due to $|S_1|/N = \hat{\alpha}$. Then, for the lower bound, we consider the same bound for $i \in S_1$ in Lemma 9.5.12 and obtain

$$\begin{aligned}\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s \rangle &= \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle + \frac{3\eta\beta_s^3}{N}\Big(\sum_{i\in S_1} \ell_i^{(t)} - \sum_{i\in S_2} \ell_i^{(t)}\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2 \\ &\geq \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle + \tilde{\Theta}(\eta)\beta_s^3\Big(\hat{\alpha}\cdot g_1(t) - \sum_{i\in S_2} \ell_i^{(t)}/N\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^2.\end{aligned}$$

□

Similarly, we have the update for the core feature as below.

Lemma 9.5.3 (Core feature update)**.** *For all $t \geq 0$ and $j \in [J]$, the core feature update is*

$$\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c \rangle = \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle + \frac{3\eta\beta_c^3}{N}\Big(\sum_{i=1}^{N} \ell_i^{(t)}\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle^2,$$

which gives

$$\begin{aligned}\tilde{\Theta}(\eta)\beta_c^3 \cdot \hat{\alpha} g_1(t) \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle^2 &\leq \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c \rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle \\ &\leq \tilde{\Theta}(\eta)\beta_c^3 \cdot \Big(\hat{\alpha} g_1(t) + \sum_{i\in S_2} \ell_i^{(t)}/N\Big) \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle^2.\end{aligned}$$

Proof. The core feature update is obtained by using the gradient update of $\mathbf{W}^{(t)}$ and plugging in (9.106):

$$\begin{aligned}\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c \rangle &= \langle \mathbf{w}_j^{(t)} - \eta \nabla_{\mathbf{w}_j} \mathcal{L}(\mathbf{W}^{(t)}), \boldsymbol{v}_c \rangle \\ &= \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle + \frac{3\eta\beta_c^3}{N}\Big(\sum_{i=1}^{N} \ell_i^{(t)}\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle^2.\end{aligned}$$

We prove for the lower bound,

$$\begin{aligned}\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c\rangle &= \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle + \frac{3\eta\beta_c^3}{N}\Big(\sum_{i=1}^{N} \ell_i^{(t)}\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2 \\ &\geq \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle + \frac{3\eta\beta_c^3}{N}\Big(\sum_{i\in S_1} \ell_i^{(t)}\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2 \\ &\geq \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle + \tilde{\Theta}(\eta)\beta_c^3\hat{\alpha}g_1(t)\cdot\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2,\end{aligned}$$

where the first inequality holds due to $0 < \ell_i^{(t)} < 1$ and the second inequality holds due to Lemma 9.5.12. And for the upper bound. we similarly have

$$\begin{aligned}\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c\rangle &= \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle + \frac{3\eta\beta_c^3}{N}\Big(\sum_{i=1}^{N} \ell_i^{(t)}\Big)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2 \\ &\leq \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle + \tilde{\Theta}(\eta)\beta_c^3\cdot\Big(\hat{\alpha}g_1(t) + \sum_{i\in S_2} \ell_i^{(t)}\Big)\cdot\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2.\end{aligned}$$

□

Note that $\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c\rangle$ is non-decreasing from the lower bound of Lemma 9.5.3. As $\mathbf{w}_j^{(0)} \sim \mathcal{N}(0, \sigma_0^2\mathbf{I}_d)$ are initialized with small σ_0, the sigmoid terms $\ell_i^{(t)}$ are large in the initial iterations. And while $l_i^{(t)}$ remains large for $i \in S_1$, we have $g_1(t) = \Theta(1)$ as similar as in [81]. Therefore, $\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s\rangle$ is also non-decreasing since $\hat{\alpha}\cdot\Theta(1) - \sum_{i\in S_2} l_i^{(t)}/N \geq 2\hat{\alpha} - 1 > 0$ for $l_i^{(t)} < 1$ and $\hat{\alpha} > 1/2$. Eventually, $g_1(t)$ becomes small at a time $T_0 > 0$. We now consider a simplified version of the above lemma in this early training stage.

Lemma 9.5.4 (Spurious feature update in early iterations)**.** *Let $T_0 > 0$ be such that $\max_{j\in[J]}\langle \mathbf{w}_j^{(T_0)}, \boldsymbol{v}_s\rangle \geq \tilde{\Omega}(1/\beta_s)$. For $t \in [0, T_0]$, the spurious feature update has the following bound*

$$\tilde{\Theta}(\eta)\beta_s^3(2\hat{\alpha}-1)\cdot\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^2 \leq \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s\rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle \leq \tilde{\Theta}(\eta)\beta_s^3\hat{\alpha}\cdot\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^2.$$

Proof. Let $T_0 > 0$ be such that either $\max_{j\in[J]}\langle \mathbf{w}_j^{(T_0)}, \boldsymbol{v}_s\rangle \geq \tilde{\Omega}(1/\beta_s)$ or $\max_{j\in[J]}\langle \mathbf{w}_j^{(T_0)}, \boldsymbol{v}_c\rangle \geq \tilde{\Omega}(1/\beta_c)$. We will show later that the first condition will be first met and we have $\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle \leq$

$\tilde{\Omega}(1/\beta_c)$ for all $j \in [J]$ and $t \in [0, T_0]$.

Recall that $g_1(t) = \text{sigmoid}\big(-\sum_{j\in[J]}(\beta_c^3\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^3 + \beta_s^3\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^3)\big)$. Then, for $t \in [0, T_0]$, we have

$$\begin{aligned} g_1(t) &= \frac{1}{1 + \exp\big(\sum_{j\in[J]}(\beta_c^3\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^3 + \beta_s^3\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^3)\big)} \\ &\geq \frac{1}{1 + \exp\big(\kappa + \kappa\big)} \\ &= \frac{1}{1 + \exp\big(\tilde{\Omega}(1)\big)}, \end{aligned}$$

where the first inequality holds due to $\langle \mathbf{w}_s^{(t)}, \boldsymbol{v}_s\rangle \leq \kappa/(J^{1/3}\beta_s)$ and $\langle \mathbf{w}_s^{(t)}, \boldsymbol{v}_c\rangle \leq \kappa/(J^{1/3}\beta_c)$ for $t \in [0, T_0]$. Therefore, similar to [81], we have $g_1(t) = \Theta(1)$ in the early iterations. Moreover, as $0 < \ell_i^{(t)} < 1$, we have $\sum_{i\in S_2}\ell_i^{(t)}/N < 1 - \hat{\alpha}$. This implies the result in Lemma 9.5.2 as

$$\tilde{\Theta}(\eta)\beta_s^3(2\hat{\alpha} - 1)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^2 \leq \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s\rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle \leq \tilde{\Theta}(\eta)\beta_s^3\hat{\alpha}\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^2.$$

□

And similarly, for the core feature, we have

Lemma 9.5.5 (Core feature update in early iterations)**.** *Let* $T_0 > 0$ *be such that* $\max_{j\in[J]}\langle \mathbf{w}_j^{(T_0)}, \boldsymbol{v}_s\rangle \geq \tilde{\Omega}(1/\beta_s)$. *For* $t \in [0, T_0]$, *the core feature update has the following bound*

$$\tilde{\Theta}(\eta)\beta_c^3\hat{\alpha} \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2 \leq \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c\rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle \leq \tilde{\Theta}(\eta)\beta_c^3 \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2.$$

Proof. Let $T_0 > 0$ be such that either $\max_{j\in[J]}\langle \mathbf{w}_j^{(T_0)}, \boldsymbol{v}_s\rangle \geq \tilde{\Omega}(1/\beta_s)$ or $\max_{j\in[J]}\langle \mathbf{w}_j^{(T_0)}, \boldsymbol{v}_c\rangle \geq \tilde{\Omega}(1/\beta_c)$. Again, with $g_1(t) = \Theta(1)$ and $\sum_{i\in S_2}\ell_i^{(t)}/N < 1 - \hat{\alpha}$ as shown in Lemma 9.5.4, we can imply the result in Lemma 9.5.3 as

$$\tilde{\Theta}(\eta)\beta_c^3\hat{\alpha} \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2 \leq \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c\rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle \leq \tilde{\Theta}(\eta)\beta_c^3 \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2,$$

which completes the proof. □

With the updates of the spurious and core feature in the early iterations, we can now show with the following lemma that GD will learn the spurious feature very quickly while hardly learning the core feature.

Lemma 9.5.6. *Let T_0 be the iteration number that $\max_{j\in[J]}\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle$ reaches $\tilde{\Omega}(1/\beta_s) = \tilde{\Theta}(1)$. Then, we have for all $t \leq T_0$, it holds that $\max_{j\in[J]}\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle = \tilde{O}(\sigma_0)$.*

Proof. Consider the following from Lemma 9.5.4 and Lemma 9.5.5,

$$\begin{aligned}\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c\rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle &\leq \tilde{\Theta}(\eta)\beta_c^3 \cdot \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2\\ \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s\rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle &\geq \tilde{\Theta}(\eta)\beta_s^3(2\hat{\alpha}-1)\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^2.\end{aligned}$$

Recall that we initialize the weights as $\mathbf{w}_j^{(0)} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2)$. We have $\langle \mathbf{w}_j^{(0)}, \boldsymbol{v}_c\rangle \sim \mathcal{N}(0, \sigma_0^2)$ and $\langle \mathbf{w}_j^{(0)}, \boldsymbol{v}_s\rangle \sim \mathcal{N}(0, \sigma_0^2)$. For the weights have small initialization with $\sigma_0 = \mathrm{polylog}(d)/d$, we have $O(\langle \mathbf{w}_j^{(0)}, \boldsymbol{v}_c\rangle) = O(\langle \mathbf{w}_j^{(0)}, \boldsymbol{v}_s\rangle)$. Therefore, for $\beta_c^3 = o(1)$ and $\beta_s^3(2\hat{\alpha}-1) = \Theta(1)$, we call Lemma 9.5.9 and get

$$\langle \mathbf{w}_j^{(T_0)}, \boldsymbol{v}_c\rangle \leq O(\langle \mathbf{w}_j^{(0)}, \boldsymbol{v}_s\rangle) = \tilde{O}(\sigma_o)$$

for all $j \in [J]$. □

Given the above lemma, we can conclude that the condition $\max_{j\in[J]}\langle \mathbf{w}_j^{(T_0)}, \boldsymbol{v}_s\rangle \geq \tilde{\Omega}(1/\beta_s)$ will be first met. And therefore, T_0 is such that $\max_{j\in[J]}\langle \mathbf{w}_j^{(T_0)}, \boldsymbol{v}_s\rangle \geq \tilde{\Omega}(1/\beta_s)$.

Theorem 9.5.7 (Restatement of Theorem 6.1.2)**.** *Consider the training dataset $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the distribution in Definition 6.1.1. Consider the two-layer nonlinear CNN model as in (6.3) initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$. After training with GD in (6.2) for $T_0 = \tilde{\Theta}\big(1/(\eta\beta_s^3\sigma_0)\big)$ iterations, for all $j \in [J]$ and $t \in [0, T_0)$, we have*

$$\tilde{\Theta}(\eta)\beta_s^3(2\hat{\alpha}-1)\cdot\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^2 \leq \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s\rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle \leq \tilde{\Theta}(\eta)\beta_s^3\hat{\alpha}\cdot\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^2, \tag{9.110}$$

$$\tilde{\Theta}(\eta)\beta_c^3\hat{\alpha}\cdot\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2 \leq \langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_c\rangle - \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle \leq \tilde{\Theta}(\eta)\beta_c^3\cdot\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle^2. \tag{9.111}$$

After training for T_0 iterations, with high probability, the learned weight has the following *properties: (1) it learns the spurious feature* $\boldsymbol{v}_s$*:* $\max_{j\in[J]}\langle \mathbf{w}_j^{(T)}, \boldsymbol{v}_s\rangle \geq \tilde{\Omega}(1/\beta_s)$*; (2) it does not learn the core feature* $\boldsymbol{v}_c$*:* $\max_{j\in[J]}\langle \mathbf{w}_j^{(T)}, \boldsymbol{v}_c\rangle = \tilde{\mathcal{O}}(\sigma_0)$*.*

Proof. The updates directly follow the results from Lemma 9.5.2 and Lemma 9.5.3. And the result for $\max_{j\in[J]}\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c\rangle$ follows Lemma 9.5.6. It remains to calculate the time T_0. With Lemma 9.5.10, we consider the sequence for $\max_{j\in[J]}\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s\rangle$, where by Lemma 9.5.4,

$$\begin{aligned}
\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s\rangle &\leq \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle + \tilde{\Theta}(\eta)\beta_s^3\hat{\alpha}\cdot\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^2,\\
\langle \mathbf{w}_j^{(t+1)}, \boldsymbol{v}_s\rangle &\geq \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle + \tilde{\Theta}(\eta)\beta_s^3(2\hat{\alpha}-1)\cdot\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle^2.
\end{aligned}$$

As $\langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s\rangle$ is non-decreasing in early iterations and with high probability, there exists an index j such that $\langle \mathbf{w}_j^{(0)}, \boldsymbol{v}_s\rangle \geq 0$. Among all the possible indices $i \in [J]$ that are initialized to have positive inner product with $\boldsymbol{v}_s$, we focus on the max index $r = \arg\max_{j\in[J]}\langle \mathbf{w}_j^{(0)}, \boldsymbol{v}_s\rangle$. Then with $v = \tilde{\Theta}(1/\beta_s)$ in Lemma 9.5.10, we will have T_0 as

$$T_0 = \frac{\tilde{\Theta}(1)}{\eta\alpha^3\sigma_0} + \frac{\tilde{\Theta}(1)\hat{\alpha}}{2\hat{\alpha}-1}\left\lceil\frac{-\log\left(\sigma_0\beta_s\right)}{\log(2)}\right\rceil.$$

□

9.5.6 Proof of Lemma 6.2.1

Lemma 9.5.8 (Restatement of Lemma 6.2.1)**.** *Given the balanced training dataset* $S^0 = \{(\mathbf{x}_i, y_i, a_i)\}_{i=1}^{N_0}$ *with* $\hat{\alpha} = 1/2$ *as in Definition 6.1.1 and CNN as in* (6.3)*. The gradient on* $\boldsymbol{v}_s$ *will be* 0 *from the beginning of training.*

Proof. With Lemma 9.5.1, the projection of the gradient on $\boldsymbol{v}_s$ in the initial iteration ($t < T_0$)

is

$$\begin{aligned}\langle \nabla_{\mathbf{w}_j}\mathcal{L}(\mathbf{W}^{(t)}), \boldsymbol{v}_s\rangle &= -\frac{3\beta_s^3}{N}\Big(\sum_{i\in S_1}\ell_i^{(t)} - \sum_{i\in S_2}\ell_i^{(t)}\Big)\cdot\langle \mathbf{w}_j, \boldsymbol{v}_s\rangle^2\\ &= \Theta\bigg(\frac{\beta_s^3}{N}\bigg)\Big(|S_1| - |S_2|\Big)\\ &= 0,\end{aligned}$$

where the first equality is due to $\ell_i^{(t)} = \Theta(1)$ in the initial iterations and the second equality is due to $\hat{\alpha} = 0.5$. □

9.5.7 Auxiliary Lemmas

Lemma 9.5.9 (Lemma C.20, **(author?)** 10). *Let $\{x_t, y_t\}_{t=1,..}$ be two positive sequences that satisfy*

$$\begin{aligned}x_{t+1} &\geq x_t + \eta\cdot Ax_t^2,\\ y_{t+1} &\leq y_t + \eta\cdot By_t^2,\end{aligned}$$

for some $A = \Theta(1)$ and $B = o(1)$. Suppose $y_0 = O(x_0)$ and $\eta < O(x_0)$, and for all $C \in [X_0, O(1)]$, let T_x be the first iteration such that $x_t \geq C$. Then, we have $T_x\eta = \Theta(x_0^{-1})$ and

$$y_{T_x} \leq O(x_0).$$

Lemma 9.5.10 (Lemma K.15, **(author?)** 81). *Let $\{z_t\}_{t=0}^T$ be a positive sequence defined by the following recursions*

$$\begin{aligned}z_{t+1} &\geq z_t + m(z_t)^2,\\ z_{t+1} &\leq z_t + M(z_t)^2,\end{aligned}$$

where $z_0 > 0$ is the initialization and $m, M > 0$ are some constants. Let $v > 0$ such that

$z_0 \leq v$. Then, the time t_0 such that $z_t \geq v$ for all $t \geq t_0$ is

$$t_0 = \frac{3}{mz_0} + \frac{8M}{m}\left\lceil \frac{\log(v/z_0)}{\log(2)} \right\rceil.$$

We make the following assumptions for every $t \leq T$ as the same in [81].

Lemma 9.5.11 (Induction hypothesis D.1, **(author?)** 81)**.** *Throughout the training process using GD for $t \leq T$, we maintain that, for every $i \in S_1$ and $j \in [J]$,*

$$|\langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle| \leq \tilde{O}(\sigma_0 \sigma \sqrt{d}). \tag{9.112}$$

Lemma 9.5.12. *For $i \in S_1$, we have $\ell_i^{(t)} = \Theta(1) g_1(t)$, where*

$$g_1(t) = sigmoid\big(- \sum_{j \in [J]} (\beta_c^3 \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_c \rangle^3 + \beta_s^3 \langle \mathbf{w}_j^{(t)}, \boldsymbol{v}_s \rangle^3)\big).$$

Proof. Given $i \in S_1$, we have from (9.109) that

$$\begin{aligned}\ell_i^{(t)} &= \text{sigmoid}\Big(\sum_{j=1}^{J} -\beta_c^3 \langle \mathbf{w}_j, \boldsymbol{v}_c \rangle^3 - \beta_s^3 \langle \mathbf{w}_j, \boldsymbol{v}_s \rangle^3 - y_i \langle \mathbf{w}_j, \boldsymbol{\xi}_i \rangle^3 \Big) \\ &= 1 \Big/ \Big(1 + \exp\Big(\sum_{j=1}^{J} \beta_c^3 \langle \mathbf{w}_j, \boldsymbol{v}_c \rangle^3 + \beta_s^3 \langle \mathbf{w}_j, \boldsymbol{v}_s \rangle^3 + y_i \langle \mathbf{w}_j, \boldsymbol{\xi}_i \rangle^3 \Big) \Big). \end{aligned} \tag{9.113}$$

Recall induction hypothesis 9.5.11, we have the following for $i \in S_1$,

$$\begin{aligned} &|y_i \langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle| \leq \tilde{O}(\sigma_0 \sigma \sqrt{d}) \\ \iff & -\tilde{O}(\sigma_0 \sigma \sqrt{d}) \leq y_i \langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle \leq \tilde{O}(\sigma_0 \sigma \sqrt{d}), \end{aligned} \tag{9.114}$$

where $|y_i| = 1$. Plug (9.114) back into (9.113), we get

$$e^{-\tilde{O}(\sigma_0 \sigma \sqrt{d})^3} g_1(t) \leq \ell_i^{(t)} \leq e^{\tilde{O}(\sigma_0 \sigma \sqrt{d})^3} g_1(t).$$

With our parameter setting, we have $\tilde{O}(\sigma_0 \sigma \sqrt{d}) = \tilde{O}(\sigma_0) = \tilde{O}(\text{polylog}(d)/d)$. Therefore,

$e^{\pm\tilde{O}(\sigma_0\sigma\sqrt{d})^3} = \Theta(1)$. □

9.6 Appendix for Chapter 7

9.6.1 Ablation Study

To better understand the impact and necessity of each loss term we conducted ablation studies for different combinations of components. Table 9.18 summarizes the findings. First, we see that choosing between the spurious image loss (row 2) or the spurious language loss (row 3) leads to similar results in worst-group accuracy but AIoU scores are higher for the spurious image loss, perhaps because decorrelation directly in the image space, but of course this requires attribute annotations. Second, the image contrastive loss is necessary for any of the spurious losses to be effective. For example, when we compare rows 2 and 3 with their corresponding versions in rows 4 and 5 that do not have the contrastive image loss, we see both worst-group acuraccy and AIoU decreasing. While this dependency on the contrastive image loss calls for more investigation, it also shows evidence that for improving vision classification results, the process of separating spurious attributes needs to happen hand in hand with separating classes from each other.

9.6.2 Limitations

While our method has demonstrated promising results in mitigating spurious correlations in ImageNet, there are still limitations to consider. We have identified several circumstances where our method may not be effective, which we discuss in more detail below.

9.6.2.1 Spurious-aware Contrastive Language Image Fine-tuning

We will describe these limitations by using the example presented in Figure 9.12. In this example, the target class we are studying is “can opener" and the spurious feature being identified by our method is the “can" itself. The spurious correlation leads to a 45.2%

Table 9.18: Investigating the impact and necessity of different loss terms with CLIP-RN50 and Waterbirds. The last four columns show the AIoU scores on the four groups (e.g. WB-L is waterbird with a land background.)

	Accuracy		AIoU		AIoU			
$\mathcal{L}_{\text{CLIP}}+$	Avg.	Worst-group	Avg.	Worst-group	LB-L	LB-W	WB-L	WB-W
(1) $\mathcal{L}_{lc}+\mathcal{L}_{vc}+\mathcal{L}_{vs}+\mathcal{L}_{ls}$	86.9%	**78.2**%	0.550	0.281	0.244	0.281	0.852	0.823
(2) $\mathcal{L}_{lc}+\mathcal{L}_{vc}+\mathcal{L}_{vs}$	83.2%	**77.5**%	**0.654**	**0.587**	0.601	0.587	0.714	0.715
(3) $\mathcal{L}_{lc}+\mathcal{L}_{vc}+\mathcal{L}_{ls}$	84.7%	**77.5**%	**0.628**	**0.499**	0.460	0.499	0.788	0.764
(4) $\mathcal{L}_{lc}+\mathcal{L}_{vs}$	83.8%	72.0%	0.525	0.310	0.329	0.310	0.762	0.697
(5) $\mathcal{L}_{lc}+\mathcal{L}_{ls}$	**88.0**%	75.1%	0.608	0.493	0.446	0.493	0.776	0.719
(6) $\mathcal{L}_{vc}+\mathcal{L}_{vs}$	86.8%	76.1%	0.609	0.461	0.441	0.461	0.768	0.763
Pre-trained CLIP	**90.8**%	44.9%	0.507	0.479	0.471	0.479	0.541	0.537
Fine-tuned CLIP	81.3%	77.1%	0.510	0.128	0.098	0.128	0.929	0.885
ERM	93.5%	54.4%	0.514	0.139	0.187	0.139	0.920	0.919
Group DRO	83.3%	73.7%	0.509	0.274	0.219	0.274	0.800	0.741

Table 9.19: Results of fine-tuning CLIP-RN50 with a subset of ImageNet classes, “can opener" and “letter opener". Both average and worst-group performance are evaluated with models early stopped at the *highest worst-group accuracy* on the validation set.

Class x	Accuracy		AIoU	
Can Opener	Avg.	Worst	Avg.	Worst
Pre-trained	86.4%	75.0%	0.568	**0.464**
Fine-tuned	76.8%	68.0%	0.561	0.339
ERM	80.1%	68.0%	0.436	0.295
Group DRO	78.4%	76.0%	0.426	0.205
Ours($\mathcal{L}_{lc}+\mathcal{L}_{vc}+\mathcal{L}_{ls}$)	73.9%	68.0%	**0.585**	0.344
Ours($\mathcal{L}_{lc}+\mathcal{L}_{vc}+\mathcal{L}_{vs}$)	75.7%	72.0%	**0.612**	**0.356**

discrepancy in accuracy and as shown in Figure 7.3, the model often focuses on the can rather than the opener.

High Concept Variation + Insufficient Data. One limitation is that the effectiveness of our method can depend on the level of variation within each class. For example, some classes may have many different variations, such as different breeds of dogs or types of flowers, while other classes may have less variation, such as types of paperclips or staplers. It is essential to note that increased variation within a class can pose a significant challenge in the learning process, even without spurious correlations. The presence of spurious correlations however can exacerbate these issues or even hide them when models are right for the wrong reasons. For example, the model can still be correct on an image with an unusual class concept variation when the spurious correlation is present (see the last two examples in Figure 9.13). Therefore, while correcting spurious correlations is vital, it alone may not address the inherent problem of object variation but the identification stage for spurious correlations itself may still be informative. Additionally, if there is insufficient data for a particular variation, our method may be unable to learn that variation effectively. For example, Figure 9.13) shows that even though our approach tends to put less emphasis on the “can" itself, it still cannot completely alleviate the problem or guarantee that the focus of the model will not shift to other parts of the image that are still spurious.

The same discussion applies to label noise, which can also be compounded by increased class variation or label subjectivity [165].

Feature Definition Challenges. Our method relies on being able to identify and isolate spurious features from non-spurious ones. However, in some cases, the spurious feature may be difficult to define or may have parts that are similar to non-spurious features. For example, in the case of a can and a can opener, the metal parts of each object may look similar and contain similar features (Figure 9.14). This can make it more challenging to separate the spurious correlations from the legitimate ones.

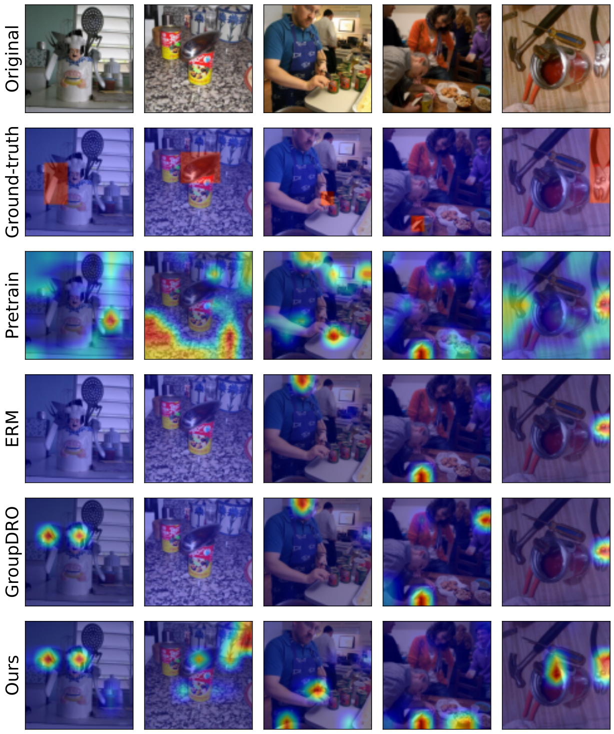


Figure 9.12: GradCAM explanations for different approaches based on CLIP RN50 for the ImageNet dataset, “can opener" class.

Due to these challenges in combination, in Table 9.19 we see that almost all methods, including ours, are not able to improve the accuracy for the “can opener" class. It is important to note that these limitations are not unique to our method, but rather are common issues that arise with spurious correlation mitigation methods and their interactions with general learning objectives. Nevertheless, the approach proposed in this work can be beneficial in many cases, in particular when significant performance drops are mainly due to spurious correlations and when spurious features do not share a visual commonality with non-spurious ones. Further research is needed to address these limitations and improve the effectiveness of spurious correlation mitigation methods in combination with other forms of out-of-distribution shifts on concept variation.

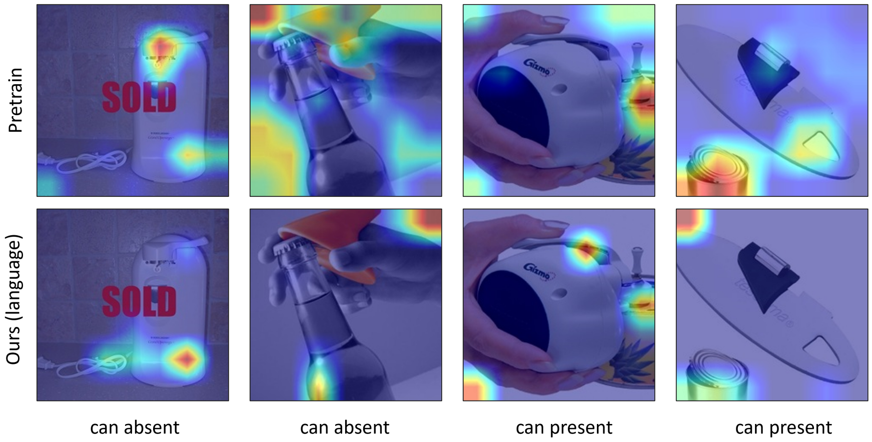


Figure 9.13: The “can opener" class in ImageNet has several concept variations, posing additional challenges in the learning process. The presence of spurious correlations can in addition exacerbate these issues or even hide them when models are right for the wrong reasons (e.g., the last two examples in this figure). While our method reduces the focus on “cans", it is still not able to completely alleviate the problem for examples with unusual concept variation.

9.6.2.2 Spurious Correlation Detection

One limitation of our spurious correlation detection method is that it relies on the object detector to identify relevant attributes. However, certain factors such as lighting and contrast may not be captured by the object detector, which can limit the effectiveness of our approach. To mitigate this limitation, our method can be used in conjunction with previous work that utilizes system/content metadata or discovered visual features for general failure analysis [138, 171, 40, 78, 57]. For instance, one can enrich the test data with additional meta-data, such as contrast, blur, lighting, and camera angle, and apply our method as well as previous approaches to understand if the model's performance drops for some of these conditions.

Note that our method leverages language attributes to describe spurious correlations but does not exclude the use of other metadata for the same purpose. While language attributes can capture a broad range of failure modes, incorporating all relevant information is crucial for comprehensive debugging.

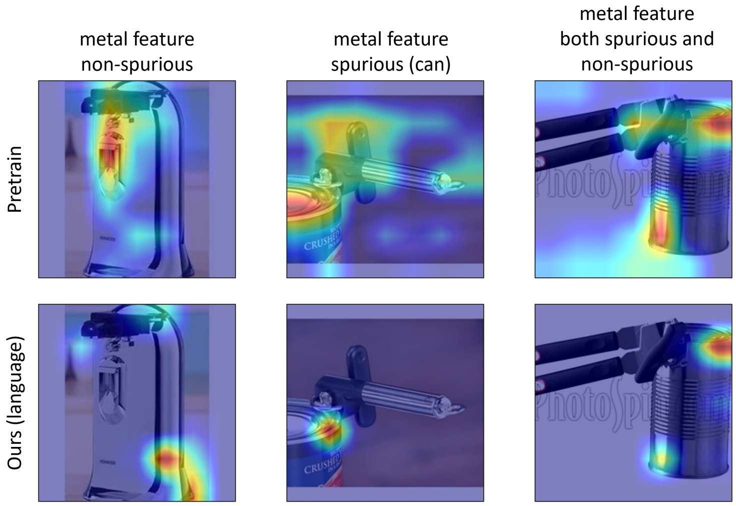


Figure 9.14: Metal features of can openers share visual commonalities with cans, which makes the problem of mitigating spurious correlations more difficult for such cases.

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pages 308–318, 2016.

[2] Amro Kamal Mohamed Abbas, Kushal Tirumala, Daniel Simig, Surya Ganguli, and Ari S Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication. In ICLR 2023 Workshop on Multimodal Representation Learning: Perks and Pitfalls, 2023.

[3] OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Benjamin Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Sim'on Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Raphael Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain,

Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Lukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Ryan Kiros, Matthew Knight, Daniel Kokotajlo, Lukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Adeola Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel P. Mossing, Tong Mu, Mira Murati, Oleg Murk, David M'ely, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Ouyang Long, Cullen O'Keefe, Jakub W. Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alexandre Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Pondé de Oliveira Pinto, Michael Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario D. Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin D. Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas A. Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cer'on Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason

Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report. 2023. URL https://api.semanticscholar.org/CorpusID:257532815.

[4] Sandhini Agarwal, Gretchen Krueger, Jack Clark, Alec Radford, Jong Wook Kim, and Miles Brundage. Evaluating clip: towards characterization of broader capabilities and downstream implications. arXiv preprint arXiv:2108.02818, 2021.

[5] Hojjat Aghakhani, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna. Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In 2021 IEEE European Symposium on Security and Privacy (EuroS&P), pages 159–178. IEEE, 2021.

[6] Faruk Ahmed, Yoshua Bengio, Harm van Seijen, and Aaron Courville. Systematic generalisation with group invariant predictions. In International Conference on Learning Representations, 2020.

[7] Faruk Ahmed, Yoshua Bengio, Harm Van Seijen, and Aaron Courville. Systematic generalisation with group invariant predictions. In International Conference on Learning Representations, 2021.

[8] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. arXiv preprint arXiv:1511.06481, 2015.

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

[10] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

[11] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[12] Hilal Asi and John C Duchi. The importance of better models in stochastic optimization. *arXiv preprint arXiv:1903.08619*, 2019.

[13] Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023.

[14] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680, 2014.

[15] Dimitri P Bertsekas. Convexification procedures and decomposition methods for nonconvex optimization problems. *Journal of Optimization Theory and Applications*, 29(2):169–197, 1979.

[16] Gantavya Bhatt, Yifang Chen, Arnav M Das, Jifan Zhang, Sang T Truong, Stephen Mussmann, Yinglun Zhu, Jeffrey Bilmes, Simon S Du, Kevin Jamieson, et al. An experimental design framework for label-efficient supervised finetuning of large language models. *arXiv preprint arXiv:2401.06692*, 2024.

[17] Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raf. Emergent and predictable memorization in large language models. *arXiv preprint arXiv:2304.11158*, 2023.

[18] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai

Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In International Conference on Machine Learning, pages 2397–2430. PMLR, 2023.

[19] 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.

[20] Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled newton methods for optimization. IMA Journal of Numerical Analysis, 39(2):545–578, 2019.

[21] 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. arXiv preprint arXiv:2108.07258, 2021.

[22] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3855–3859. IEEE, 2021.

[23] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://aclanthology.org/D15-1075.

[24] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

[25] 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, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[26] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[27] Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In International Conference on Machine Learning, pages 872–881. PMLR, 2019.

[28] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. Advances in neural information processing systems, 32, 2019.

[29] Kaidi Cao, Yining Chen, Junwei Lu, Nikos Arechiga, Adrien Gaidon, and Tengyu Ma. Heteroskedastic and imbalanced deep learning with adaptive regularization. arXiv preprint arXiv:2006.15766, 2020.

[30] Yuan Cao, Zixiang Chen, Misha Belkin, and Quanquan Gu. Benign overfitting in two-layer convolutional neural networks. Advances in neural information processing systems, 35:25237–25250, 2022.

[31] Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Accelerated methods for nonconvex optimization. SIAM Journal on Optimization, 28(2):1751–1772, 2018.

[32] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In SafeAI@ AAAI, 2019.

[33] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=FdVXgSJhvz.

[34] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526, 2017.

[35] Yining Chen, Colin Wei, Ananya Kumar, and Tengyu Ma. Self-training avoids using spurious features under domain shift. arXiv preprint arXiv:2006.10032, 2020.

[36] Yongqiang Chen, Wei Huang, Kaiwen Zhou, Yatao Bian, Bo Han, and James Cheng. Towards understanding feature learning in out-of-distribution generalization. arXiv preprint arXiv:2304.11327, 2023.

[37] Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. Towards understanding the mixture-of-experts layer in deep learning. Advances in neural information processing systems, 2022.

[38] Daixuan Cheng, Shaohan Huang, and Furu Wei. Adapting large language models via reading comprehension. arXiv preprint arXiv:2309.09530, 2023.

[39] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. Journal of Machine Learning Research, 24(240):1–113, 2023.

[40] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. Slice finder: Automated data slicing for model validation. In 2019 IEEE 35th International Conference on Data Engineering (ICDE), pages 1550–1553. IEEE, 2019.

[41] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

[42] 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.

[43] Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In International Conference on Machine Learning, pages 2189–2200. PMLR, 2021.

[44] Gabriela F Cretu, Angelos Stavrou, Michael E Locasto, Salvatore J Stolfo, and Angelos D Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In 2008 IEEE Symposium on Security and Privacy (sp 2008), pages 81–95. IEEE, 2008.

[45] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 9268–9277, 2019.

[46] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. Nature, 600(7887):70–74, 2021.

[47] Aaron Defazio and Leon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. Advances in Neural Information Processing Systems, 32:1755–1765, 2019.

[48] Jean-Benoit Delbrouck, Maya Varma, Pierre Chambon, and Curtis Langlotz. Overview of the RadSum23 shared task on multi-modal and multi-anatomical radiology report summarization. In Dina Demner-fushman, Sophia Ananiadou, and Kevin Cohen, editors, The 22nd Workshop on Biomedical Natural Language Processing and BioNLP Shared Tasks, pages 478–482, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.bionlp-1.45. URL https://aclanthology.org/2023.bionlp-1.45.

[49] Ron S Dembo, Stanley C Eisenstat, and Trond Steihaug. Inexact newton methods. SIAM Journal on Numerical analysis, 19(2):400–408, 1982.

[50] Dina Demner-fushman, Sophia Ananiadou, and Kevin Cohen, editors. The 22nd Workshop on Biomedical Natural Language Processing and BioNLP Shared Tasks, Toronto, Canada, July 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.bionlp-1.0.

[51] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

[52] Yihe Deng, Yu Yang, Baharan Mirzasoleiman, and Quanquan Gu. Robust learning with progressive data expansion against spurious correlation. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=9QEVJ9qm46.

[53] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[54] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

[55] John C Duchi, Tatsunori Hashimoto, and Hongseok Namkoong. Distributionally robust losses against mixture covariate shifts. Under review, 2:1, 2019.

[56] Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? arXiv preprint arXiv:2305.07759, 2023.

[57] Sabri Eyuboglu, Maya Varma, Khaled Kamal Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Re. Domino: Discovering systematic errors with cross-modal embeddings. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=FPCMqjI0jXN.

[58] 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. Advances in Neural Information Processing Systems, 33:5850–5861, 2020.

[59] Jonas Geiping, Liam Fowl, Gowthami Somepalli, Micah Goldblum, Michael Moeller, and Tom Goldstein. What doesn't kill you makes you robust (er): Adversarial training against poisons and backdoors. arXiv preprint arXiv:2102.13624, 2021.

[60] Jonas Geiping, Liam H Fowl, W. Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=01olnfLIbD.

[61] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. SIAM Journal on Optimization, 23(4):2341–2368, 2013.

[62] Karan Goel, Albert Gu, Yixuan Li, and Christopher Re. Model patching: Closing the subgroup performance gap with data augmentation. In International Conference on Learning Representations, 2021.

[63] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677, 2017.

[64] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733, 2017.

[65] Nika Haghtalab, Michael Jordan, and Eric Zhao. On-demand sampling: Learning optimally from multiple distributions. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, Advances in Neural Information Processing Systems, 2022.

[66] Haibo He and Edwardo A Garcia. Learning from imbalanced data. IEEE Transactions on knowledge and data engineering, 21(9):1263–1284, 2009.

[67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[68] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

[69] Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. Advances in Neural Information Processing Systems, 33:9995–10006, 2020.

[70] Sanghyun Hong, Varun Chandrasekaran, Yiğitcan Kaya, Tudor Dumitraş, and Nicolas Papernot. On the effectiveness of mitigating data poisoning attacks with gradient shaping. arXiv preprint arXiv:2002.11497, 2020.

[71] Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. The surprising simplicity of the early-time learning dynamics of neural networks. Advances in Neural Information Processing Systems, 33:17116–17128, 2020.

[72] Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. Does distributionally robust supervised learning give robust classifiers? In International Conference on Machine Learning, pages 2029–2037. PMLR, 2018.

[73] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708, 2017.

[74] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. Advances in Neural Information Processing Systems, 33, 2020.

[75] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. Communications in Statistics-Simulation and Computation, 18(3):1059–1076, 1989.

[76] Pavel Izmailov, Polina Kirichenko, Nate Gruver, and Andrew G Wilson. On feature learning in the presence of spurious correlations. Advances in Neural Information Processing Systems, 35:38516–38532, 2022.

[77] Pavel Izmailov, Polina Kirichenko, Nate Gruver, and Andrew Gordon Wilson. On feature learning in the presence of spurious correlations. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, Advances in Neural Information Processing Systems, 2022. URL https://openreview.net/forum?id=wKhUPzqVap6.

[78] Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space. arXiv preprint arXiv:2206.14754, 2022.

[79] Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Exploring the benefits of training

expert language models over instruction tuning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pages 14702–14729. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/jang23a.html.

[80] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In 28th {USENIX} Security Symposium ({USENIX} Security 19), pages 1895–1912, 2019.

[81] Samy Jelassi and Yuanzhi Li. Towards understanding how momentum improves generalization in deep learning. In International Conference on Machine Learning, pages 9965–10040. PMLR, 2022.

[82] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I Jordan. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. Journal of the ACM (JACM), 68(2):1–29, 2021.

[83] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. Scientific data, 3 (1):1–9, 2016.

[84] Siddharth Joshi and Baharan Mirzasoleiman. Data-efficient contrastive self-supervised learning: Most beneficial examples for supervised learning contribute the least. In International conference on machine learning, pages 15356–15370. PMLR, 2023.

[85] Siddharth Joshi, Yu Yang, Yihao Xue, Wenhan Yang, and Baharan Mirzasoleiman. Towards mitigating spurious correlations in the wild: A benchmark & a more realistic dataset. arXiv preprint arXiv:2306.11957, 2023.

[86] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding.

In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 1780–1790, 2021.

[87] Angelos Katharopoulos and Francois Fleuret. Not all samples are created equal: Deep learning with importance sampling. In International Conference on Machine Learning, pages 2525–2534, 2018.

[88] Krishnateja Killamsetty, S Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In International Conference on Machine Learning, pages 5464–5474. PMLR, 2021.

[89] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 8110–8118, 2021.

[90] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=Zb6c8A-Fghk.

[91] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. arXiv preprint arXiv:1811.00741, 2018.

[92] 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.

[93] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In International Conference on Machine Learning, pages 5637–5664. PMLR, 2021.

[94] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1152–1157, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1136. URL https://aclanthology.org/N16-1136.

[95] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016.

[96] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[97] Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defenses against general poisoning attacks. In International Conference on Learning Representations, 2020.

[98] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. Advances in Neural Information Processing Systems, 35:3843–3857, 2022.

[99] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu

Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10965–10975, 2022.

[100] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. Advances in Neural Information Processing Systems, 34, 2021.

[101] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. arXiv preprint arXiv:2309.05463, 2023.

[102] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: **phi-1.5** technical report. arXiv preprint arXiv:2309.05463, 2023.

[103] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

[104] Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. Tinygsm: achieving> 80% on gsm8k with small language models. arXiv preprint arXiv:2312.09241, 2023.

[105] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Toward a theory of optimization for over-parameterized systems of non-linear equations: the lessons of deep learning. arXiv preprint arXiv:2003.00307, 2020.

[106] Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In International Conference on Machine Learning, pages 6781–6792. PMLR, 2021.

[107] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

[108] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[109] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of the IEEE international conference on computer vision, pages 3730–3738, 2015.

[110] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

[111] Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. arXiv preprint arXiv:1511.06343, 2015.

[112] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. arXiv preprint arXiv:2308.09583, 2023.

[113] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. arXiv preprint arXiv:2306.08568, 2023.

[114] Yuzhe Ma, Xiaojin Zhu Zhu, and Justin Hsu. Data poisoning against differentially-private learners: Attacks and defenses. In International Joint Conference on Artificial Intelligence, 2019.

[115] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representations, 2018.

[116] Anas Mahmoud, Mostafa Elhoushi, Amro Abbas, Yu Yang, Newsha Ardalani, Hugh Leather, and Ari Morcos. Sieve: Multimodal dataset pruning using image-captioning models. In Conference on Computer Vision and Pattern Recognition, 2024. URL https://openreview.net/forum?id=DBxBPGRWjw.

[117] Chengzhi Mao, Kevin Xia, James Wang, Hao Wang, Junfeng Yang, Elias Bareinboim, and Carl Vondrick. Causal transportability for visual recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7521–7531, 2022.

[118] Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining llms at scale. arXiv preprint arXiv:2309.04564, 2023.

[119] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. Journal of the society for Industrial and Applied Mathematics, 11(2):431–441, 1963.

[120] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In International conference on machine learning, pages 2408–2417. PMLR, 2015.

[121] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection with vision transformers. arXiv preprint arXiv:2205.06230, 2022.

[122] Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In International Conference on Machine Learning, pages 15630–15649. PMLR, 2022.

[123] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In Optimization techniques, pages 234–243. Springer, 1978.

[124] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In Advances in Neural Information Processing Systems, pages 2049–2057, 2013.

[125] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[126] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. ICML, 2020.

[127] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In Hal Daumé III and Aarti Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 6950–6960. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/mirzasoleiman20a.html.

[128] Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. NumGLUE: A suite of fundamental yet challenging mathematical reasoning tasks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3505–3523, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.246. URL https://aclanthology.org/2022.acl-long.246.

[129] Mazda Moayeri, Phillip Pope, Yogesh Balaji, and Soheil Feizi. A comprehensive study of image classification model sensitivity to foregrounds, backgrounds, and visual attributes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 19087–19097, 2022.

[130] Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. arXiv preprint arXiv:2010.15775, 2020.

[131] Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 3496–3506, 2019.

[132] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: De-biasing classifier from biased classifier. Advances in Neural Information Processing Systems, 33:20673–20684, 2020.

[133] Junhyun Nam, Jaehyung Kim, Jaeho Lee, and Jinwoo Shin. Spread spurious attribute: Improving worst-group accuracy with spurious attribute estimation. In International Conference on Learning Representations, 2021.

[134] Junhyun Nam, Jaehyung Kim, Jaeho Lee, and Jinwoo Shin. Spread spurious attribute: Improving worst-group accuracy with spurious attribute estimation. arXiv preprint arXiv:2204.02070, 2022.

[135] Hongseok Namkoong and John C Duchi. Variance-based regularization with convex objectives. Advances in neural information processing systems, 30, 2017.

[136] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. arXiv preprint arXiv:1412.6614, 2014.

[137] Giang Nguyen, Daeyoung Kim, and Anh Nguyen. The effectiveness of feature attribution methods and its correlation with automatic evaluation scores. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, Advances in Neural Information Processing Systems, 2021. URL https://openreview.net/forum?id=OKPS9YdZ8Va.

[138] Besmira Nushi, Ece Kamar, and Eric Horvitz. Towards accountable ai: Hybrid human-machine analyses for characterizing system failure. In Proceedings of the AAAI Conference on Human Computation and Crowdsourcing, volume 6, pages 126–135, 2018.

[139] Yonatan Oren, Shiori Sagawa, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust language modeling. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4227–4237, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1432.

[140] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pages 311–318, 2002.

[141] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. 2019.

[142] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168.

[143] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data

diet: Finding important examples early in training. Advances in Neural Information Processing Systems, 34, 2021.

[144] Neehar Peri, Neal Gupta, W Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P Dickerson. Deep k-nn defense against clean-label data poisoning attacks. In European Conference on Computer Vision, pages 55–70. Springer, 2020.

[145] Suzanne Petryk, Lisa Dunlap, Keyan Nasseri, Joseph Gonzalez, Trevor Darrell, and Anna Rohrbach. On guiding visual attention with language specification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 18092–18102, 2022.

[146] Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. Advances in Neural Information Processing Systems, 34:1256–1272, 2021.

[147] Omead Pooladzandi, David Davini, and Baharan Mirzasoleiman. Adaptive second order coresets for data-efficient machine learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 17848–17869. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/pooladzandi22a.html.

[148] Omead Pooladzandi, David Davini, and Baharan Mirzasoleiman. Adaptive second order coresets for data-efficient machine learning. In International Conference on Machine Learning, pages 17848–17869. PMLR, 2022.

[149] Neha Prakriya, Yu Yang, Baharan Mirzasoleiman, Cho-Jui Hsieh, and Jason Cong. Nessa: Near-storage data selection for accelerated machine learning training. In Proceedings of the 15th ACM Workshop on Hot Topics in Storage and File Systems, HotStorage '23, page 8–15, New York, NY, USA, 2023. Association for Computing

Machinery. ISBN 9798400702242. doi: 10.1145/3599691.3603404. URL https://doi.org/10.1145/3599691.3603404.

[150] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In International Conference on Machine Learning, pages 8748–8763. PMLR, 2021.

[151] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20:53–65, 1987.

[152] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. arXiv preprint arXiv:2308.12950, 2023.

[153] 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), 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[154] 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), 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[155] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks. In International Conference on Learning Representations, 2019.

[156] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In International Conference on Machine Learning, pages 8346–8356. PMLR, 2020.

[157] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks, 2019.

[158] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4510–4520, 2018.

[159] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015.

[160] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. arXiv preprint arXiv:1907.10597, 2019.

[161] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. arXiv preprint arXiv:2006.12557, 2020.

[162] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision, pages 618–626, 2017.

[163] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks, 2018.

[164] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. Advances in Neural Information Processing Systems, 33:9573–9585, 2020.

[165] Vaishaal Shankar, Rebecca Roelofs, Horia Mania, Alex Fang, Benjamin Recht, and Ludwig Schmidt. Evaluating machine accuracy on ImageNet. In Hal Daumé III and Aarti Singh, editors, Proceedings of the 37th International Conference on Machine Learning,

volume 119 of Proceedings of Machine Learning Research, pages 8634–8644. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/shankar20c.html.

[166] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. Journal of statistical planning and inference, 90(2):227–244, 2000.

[167] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[168] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Workshop at International Conference on Learning Representations, 2014.

[169] Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. Nature, 620(7972):172–180, 2023.

[170] Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. Towards expert-level medical question answering with large language models. arXiv preprint arXiv:2305.09617, 2023.

[171] Sahil Singla, Besmira Nushi, Shital Shah, Ece Kamar, and Eric Horvitz. Understanding failures of deep networks via robust feature extraction. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021. Computer Vision Foundation / IEEE, 2021.

[172] Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. Advances in Neural Information Processing Systems, 33:19339–19352, 2020.

[173] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos.

Beyond neural scaling laws: beating power law scaling via data pruning. Advances in Neural Information Processing Systems, 35:19523–19536, 2022.

[174] Hossein Souri, Micah Goldblum, Liam Fowl, Rama Chellappa, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. arXiv preprint arXiv:2106.08970, 2021.

[175] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks, 2017.

[176] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:1906.02243, 2019.

[177] Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 9275–9293, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.746. URL https://aclanthology.org/2020.emnlp-main.746.

[178] Saeid A Taghanaki, Kristy Choi, Amir Hosein Khasahmadi, and Anirudh Goyal. Robust representation learning via perceptual similarity metrics. In International Conference on Machine Learning, pages 10043–10053. PMLR, 2021.

[179] Lue Tao, Lei Feng, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. Better safe than sorry: Preventing delusive adversaries with adversarial training. Advances in Neural Information Processing Systems, 34, 2021.

[180] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. arXiv preprint arXiv:2211.09085, 2022.

[181] Damien Teney, Ehsan Abbasnejad, Simon Lucey, and Anton Van den Hengel. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior ood generalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16761–16772, 2022.

[182] Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari S. Morcos. D4: Improving LLM pretraining via document de-duplication and diversification. In Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2023. URL https://openreview.net/forum?id=CG0L2PFrb1.

[183] 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, 2018.

[184] 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, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

[185] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

[186] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In Advances in Neural Information Processing Systems, pages 8000–8010, 2018.

[187] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=SyxAb30cY7.

[188] Tao Tu, Shekoofeh Azizi, Danny Driess, Mike Schaekermann, Mohamed Amin, Pi-Chuan Chang, Andrew Carroll, Chuck Lau, Ryutaro Tanno, Ira Ktena, et al. Towards generalist biomedical ai. arXiv preprint arXiv:2307.14334, 2023.

[189] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018.

[190] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(11), 2008.

[191] Dave Van Veen, Cara Van Uden, Louis Blankemeier, Jean-Benoit Delbrouck, Asad Aali, Christian Bluethgen, Anuj Pareek, Malgorzata Polacin, William Collins, Neera Ahuja, et al. Clinical text summarization: adapting large language models can outperform human experts. arXiv preprint arXiv:2309.07430, 2023.

[192] Neeraj Varshney, Swaroop Mishra, and Chitta Baral. Let the model decide its curriculum for multitask learning. In Colin Cherry, Angela Fan, George Foster, Gholamreza (Reza) Haffari, Shahram Khadivi, Nanyun (Violet) Peng, Xiang Ren, Ehsan Shareghi, and Swabha Swayamdipta, editors, Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing, pages 117–125, Hybrid, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.deeplo-1.13. URL https://aclanthology.org/2022.deeplo-1.13.

[193] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

[194] Akshaj Veldanda and Siddharth Garg. On evaluating neural network backdoor defenses. arXiv preprint arXiv:2010.12186, 2020.

[195] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[196] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[197] Weiran Wang and Nathan Srebro. Stochastic nonconvex optimization with large minibatches. In Algorithmic Learning Theory, pages 857–882. PMLR, 2019.

[198] Maurice Weber, Xiaojun Xu, Bojan Karlaš, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks. arXiv preprint arXiv:2003.08904, 2020.

[199] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, Advances in Neural Information Processing Systems, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

[200] Jiaheng Wei, Harikrishna Narasimhan, Ehsan Amid, Wen-Sheng Chu, Yang Liu, and Abhishek Kumar. Distributionally robust post-hoc classifiers under prior shifts. In International Conference on Learning Representations (ICLR), 2023.

[201] Zixin Wen and Yuanzhi Li. Toward understanding the feature learning process of self-supervised contrastive learning. In International Conference on Machine Learning, pages 11112–11122. PMLR, 2021.

[202] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[203] Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica, 2(4):385–393, 1982.

[204] Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. Self-evolved diverse data sampling for efficient instruction tuning. arXiv preprint arXiv:2311.08182, 2023.

[205] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. arXiv preprint arXiv:2303.17564, 2023.

[206] Mengzhou Xia, Mikel Artetxe, Chunting Zhou, Xi Victoria Lin, Ramakanth Pasunuru, Danqi Chen, Luke Zettlemoyer, and Veselin Stoyanov. Training trajectories of language models across scales. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13711–13738, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.767. URL https://aclanthology.org/2023.acl-long.767.

[207] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2691–2699, 2015.

[208] Da Xu, Yuting Ye, and Chuanwei Ruan. Understanding the role of importance weighting for deep learning. In International Conference on Learning Representations, 2021.

[209] Peng Xu, Fred Roosta, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. Mathematical Programming, 184(1): 35–70, 2020.

[210] Yihao Xue, Ali Payani, Yu Yang, and Baharan Mirzasoleiman. Eliminating spurious correlations from pre-trained models via data mixing. arXiv preprint arXiv:2305.14521, 2023.

[211] Yao-Yuan Yang, Chi-Ning Chou, and Kamalika Chaudhuri. Understanding rare spurious correlations in neural networks. arXiv preprint arXiv:2202.05189, 2022.

[212] Yu Yang, Seungbae Kim, and Jungseock Joo. Explaining deep convolutional neural networks via latent visual-semantic filter attention. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8333–8343, 2022.

[213] Yu Yang, Tian Yu Liu, and Baharan Mirzasoleiman. Not all poisons are created equal: Robust training against data poisoning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 25154–25165. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/yang22j.html.

[214] Yu Yang, Hao Kang, and Baharan Mirzasoleiman. Towards sustainable learning: Coresets for data-efficient deep learning. In Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pages 39314–39330. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/yang23g.html.

[215] Yu Yang, Besmira Nushi, Hamid Palangi, and Baharan Mirzasoleiman. Mitigating spurious correlations in multi-modal models during fine-tuning. In International Conference on Machine Learning, 2023.

[216] Yu Yang, Aaditya K Singh, Mostafa Elhoushi, Anas Mahmoud, Kushal Tirumala, Fabian Gloeckle, Baptiste Rozière, Carole-Jean Wu, Ari S Morcos, and Newsha Ardalani. Decoding data quality via synthetic corruptions: Embedding-guided pruning of code data. arXiv preprint arXiv:2312.02418, 2023.

[217] Yu Yang, Eric Gan, Gintare Karolina Dziugaite, and Baharan Mirzasoleiman. Identifying spurious biases early in training through the lens of simplicity bias. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, Proceedings of The

27th International Conference on Artificial Intelligence and Statistics, volume 238 of Proceedings of Machine Learning Research, pages 2953–2961. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/yang24c.html.

[218] Yuzhe Yang, Haoran Zhang, Dina Katabi, and Marzyeh Ghassemi. Change is hard: A closer look at subpopulation shift. arXiv preprint arXiv:2302.12254, 2023.

[219] Zhewei Yao, Peng Xu, Farbod Roosta-Khorasani, and Michael W Mahoney. Inexact non-convex newton-type methods. arXiv preprint arXiv:1802.06925, 2018.

[220] Zhewei Yao, Amir Gholami, Sheng Shen, Kurt Keutzer, and Michael W Mahoney. Adahessian: An adaptive second order optimizer for machine learning. arXiv preprint arXiv:2006.00719, 2020.

[221] Haotian Ye, James Zou, and Linjun Zhang. Freeze then train: Towards provable representation learning under spurious correlations and feature noise. arXiv preprint arXiv:2210.11075, 2022.

[222] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In ICML Deep Learning Workshop, 2016.

[223] Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

[224] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building math generalist models through hybrid instruction tuning. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=yLClGs770I.

[225] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In British Machine Vision Conference 2016. British Machine Vision Association, 2016.

[226] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12104–12113, 2022.

[227] Jingzhao Zhang, Aditya Krishna Menon, Andreas Veit, Srinadh Bhojanapalli, Sanjiv Kumar, and Suvrit Sra. Coping with label shift via distributionally robust optimisation. In International Conference on Learning Representations, 2021.

[228] Michael Zhang and Christopher Re. Contrastive adapters for foundation model group robustness. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, Advances in Neural Information Processing Systems, 2022. URL https://openreview.net/forum?id=uPdS_7pdA9p.

[229] Michael Zhang, Nimit S Sohoni, Hongyang R Zhang, Chelsea Finn, and Christopher Re. Correct-n-contrast: a contrastive approach for improving robustness to spurious correlations. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 26484–26516. PMLR, 17–23 Jul 2022.

[230] Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. Interpreting cnns via decision trees. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 6261–6270, 2019.

[231] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

[232] Yuhui Zhang, Jeff Z. HaoChen, Shih-Cheng Huang, Kuan-Chieh Wang, James Zou, and Serena Yeung. Diagnosing and rectifying vision models using language. In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=D-zfUK7BR6c.

[233] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. arXiv preprint arXiv:2304.11277, 2023.

[234] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[235] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. LIMA: Less is more for alignment. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=KBMOKmX2he.

[236] Haotian Zhou, Tingkai Liu, Qianli Ma, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. Lobass: Gauging learnability in supervised fine-tuning data. arXiv preprint arXiv:2310.13008, 2023.

[237] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. Proceedings of the AAAI Conference on Artificial Intelligence, 34(07):13041–13049, 2020.

[238] Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In International Conference on Machine Learning, pages 7614–7623, 2019.

[239] Difan Zou, Yuan Cao, Yuanzhi Li, and Quanquan Gu. Understanding the generaliza-

tion of adam in learning neural networks with proper regularization. arXiv preprint arXiv:2108.11371, 2021.