UNIVERSITY OF CALIFORNIA SAN DIEGO

Training Data-Constrained Language Models

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

in

Computer Science

by

Chengyu Dong

Committee in charge:

Jingbo Shang, Chair
Mikhail Belkin
Taylor Berg-Kirkpatrick
Xiaodong Liu
Julian McAuley

2025

The Dissertation of Chengyu Dong is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2025

EPIGRAPH

More Is Different.

Philip Warren Anderson

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to express my heartfelt gratitude to my PhD advisor, Professor Jingbo Shang, for his unwavering support throughout my PhD journey. His insightful guidance and mentorship not only shaped me into a confident researcher in the field of artificial intelligence but also helped me grow immensely on a personal level.

My sincere thanks also go to Dr. Mikhail Belkin, Dr. Taylor Berg-Kirkpatrick, Dr. Xiaodong Liu, and Dr. Julian McAuley for serving on my thesis committee. Their valuable advice and thoughtful feedback have been instrumental in improving my dissertation and preparing me for a successful defense.

I am sincerely thankful to my mentors and collaborators from Microsoft Research, Google DeepMind, and Nvidia—particularly Liyuan Liu, Hao Cheng, Huan Gui, Zhe Zhao, and Mostofa Patwary. Interning at these prestigious institutions has been an incredibly rewarding experience, and their mentorship and encouragement have directly contributed to the fruitful outcomes of my internships.

I extend my deepest appreciation to my labmates in the Shang Data Lab, as well as our sibling labs—the Julian Lab and the Zhiting Lab. Together, we shared an incredible five-year journey filled with mutual support, stimulating discussions, constructive feedback, and creative brainstorming sessions. Beyond research, our friendships flourished through dinners, board game nights, and pick-up sports, bringing joy, relief, and renewed energy to our busy lives.

Special thanks go to my dear friends in Soccer Illuminati. Our weekend soccer matches were always among the happiest and most anticipated moments of my week. Likewise, I am profoundly grateful to my friends in the Fusheng Chinese Drama Troupe. Acting on stage had been a dream of mine since childhood, and each performance under the dazzling magnesium lights and before hundreds of attentive audience members brought a new dimension of joy and fulfillment to my life.

Finally, and most importantly, I would like to express my deepest gratitude to my wife and my parents. Their unwavering understanding, encouragement, and love have consistently

been my source of strength during moments of stress and anxiety. They have inspired and motivated me every step of the way, empowering me to reach higher and achieve my goals.

Chapter 1 incorporates material from the publication "Understand and Modularize Generator Optimization in ELECTRA-style Pretraining" by Chengyu Dong, Liyuan Liu, Hao Cheng, Jingbo Shang, Jianfeng Gao, Xiaodong Liu, published in Proceedings of the 40th International Conference on Machine Learning (ICML 2023). The dissertation author was the primary investigator and lead author of this paper.

Chapter 2 incorporates material from the publication "Toward Student-oriented Teacher Network Training for Knowledge Distillation" by Chengyu Dong, Liyuan Liu, and Jingbo Shang, published in The Twelfth International Conference on Learning Representations (ICLR 2024). The dissertation author was the primary investigator and lead author of this paper.

Chapter 3 incorporates material from the publication "Debiasing Made State-of-the-art: Revisiting the Simple Seed-based Weak Supervision for Text Classification" by Chengyu Dong, Zihan Wang, and Jingbo Shang, published in The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023). The dissertation author was the primary investigator and lead author of this paper.

VITA

2016	Bachelor of Science in Astronomy, Nanjing University
2018	Master of Science in Astrometry & Celestial Mechanics, Nanjing University
2020	Master of Science in Computer Science, University of California San Diego
2020–2021	Research Assistant, University of California San Diego
2025	Doctor of Philosophy in Computer Science, University of California San Diego

PUBLICATIONS

[1] Alex Nguyen, Dheeraj Mekala, Chengyu Dong, Jingbo Shang. When is the Consistent Prediction Likely to Be A Correct Prediction?

[2] Chengyu Dong, Liyuan Liu and Jingbo Shang. Toward Student-Oriented Teacher Network Training For Knowledge Distillation. ICLR 2024.

[3] Chengyu Dong, Liyuan Liu, Hao Cheng, Jingbo Shang, Jianfeng Gao and Xiaodong Liu. Fast-ELECTRA for Efficient Pre-training. ICLR 2024.

[4] Liyuan Liu, Chengyu Dong, Xiaodong Liu, Bin Yu, Jianfeng Gao. Bridging Discrete and Backpropagation: Straight-Through and Beyond. NeurIPS 2023 (Oral).

[5] Chengyu Dong, Zihan Wang, Jingbo Shang. Debiasing Made State-of-the-art: Revisiting the Simple Seed-based Weak Supervision for Text Classification. EMNLP 2023.

[6] Chengyu Dong, Liyuan Liu, Hao Cheng, Jingbo Shang, Jianfeng Gao and Xiaodong Liu. Understand and Modularize Generator Optimization in ELECTRA-style Pretraining. ICML 2023.

[7] Chengyu Dong, Liyuan Liu and Jingbo Shang. Label Noise in Adversarial Training: A Novel Perspective to Study Robust Overfitting. NeurIPS 2022 (Oral).

[8] Chengyu Dong, Liyuan Liu and Jingbo Shang. Data Quality Matters For Adversarial Training: An Empirical Study. arXiv:2102.07437.

[9] Chengyu Dong, Liyuan Liu and Jingbo Shang. Towards Adaptive Residual Network Training: A Neural-ODE Perspective. ICML 2020.

[10] An Yan, Yu Wang, Yiwu Zhong, Chengyu Dong, Zexue He, Yujie Lu, William Yang Wang, Jingbo Shang, Julian McAuley. Learning concise and descriptive attributes for visual recognition.

ICCV 2023.

[11] An Yan, Yu Wang, Yiwu Zhong, Zexue He, Petros Karypis, Zihan Wang, Chengyu Dong, Amilcare Gentili, Chun-Nan Hsu, Jingbo Shang, Julian McAuley. Robust and interpretable medical image classifiers via concept bottleneck models.

[12] Zichao Li, Dheeraj Mekala, Chengyu Dong, Jingbo Shang. Bfclass: A backdoor-free text classification framework. EMNLP 2021.

[13] Dheeraj Mekala, Chengyu Dong, Jingbo Shang. LOPS: Learning order inspired pseudo-label selection for weakly supervised text classification. EMNLP 2022.

[14] Xiyuan Zhang, Xiaohan Fu, Diyan Teng, Chengyu Dong, Keerthivasan Vijayakumar, Jiayun Zhang, Ranak Roy Chowdhury, Junsheng Han, Dezhi Hong, Rashmi Kulkarni, Jingbo Shang, Rajesh K Gupta. Physics-informed data denoising for real-life sensing systems. SenSys 2023.

[15] Dheeraj Mekala, Adithya Samavedhi, Chengyu Dong, Jingbo Shang. SELFOOD: Self-Supervised Out-Of-Distribution Detection via Learning to Rank. EMNLP 2023.

[16] Shang Zhou, Feng Yao, Chengyu Dong, Zihan Wang, Jingbo Shang. Evaluating the Smooth Control of Attribute Intensity in Text Generation with LLMs. ACL 2024 Findings.

[17] Letian Peng, Chenyang An, Shibo Hao, Chengyu Dong, Jingbo Shang. Linear Correlation in LM's Compositional Generalization and Hallucination.

[18] Chenyang An, Shima Imani, Feng Yao, Chengyu Dong, Ali Abbasi, Harsh Shrivastava, Samuel Buss, Jingbo Shang, Gayathri Mahalingam, Pramod Sharma, Maurice Diesendruck. Next-Token Prediction Task Assumes Optimal Data Ordering for LLM Training in Proof Generation.

[19] Letian Peng, Yi Gu, Chengyu Dong, Zihan Wang, Jingbo Shang. Text Grafting: Near-Distribution Weak Supervision for Minority Classes in Text Classification. EMNLP 2024.

ABSTRACT OF THE DISSERTATION

Training Data-Constrained Language Models

by

Chengyu Dong

Doctor of Philosophy in Computer Science

University of California San Diego, 2025

Jingbo Shang, Chair

The rapid advancement of language models has significantly reshaped the field of machine learning, enabling sophisticated applications across diverse domains. However, the effectiveness of these models is often contingent upon access to large-scale, high-quality datasets, which may not always be available. This thesis explores strategies for training language models in data-constrained scenarios, leveraging both model-generated data and rule-generated data to enhance model performance and generalization.

First, we investigate ELECTRA, a pretraining framework that improves the data efficiency of language model training by utilizing an "adversarial" model to corrupt the training data. Through theoretical and empirical analysis, we identify a critical optimization control problem

in the original ELECTRA design, and propose a simple fix to boost its data efficiency further.

Next, we study knowledge distillation, which improves model training by utilizing a "supportive" teacher model to refine the training data. Through theoretical analysis, we demonstrate the effectiveness of knowledge distillation even when the teacher model is trained with the exact same training set as the student model. Motivated by our understanding, we develop a student-oriented teacher training framework that specifically optimizes the teacher model to maximize student performance, rather than its own accuracy.

Finally, we investigate rule-generated data. We focus on weakly-supervised learning, which leverages heuristics to automatically and scalably annotate datasets without human annotation. Through empirical analyses, we identify a key problem of model training on such data labeled by rules - the model is easily biased by the simple heuristics used to annotate the data. We design a simple method to avoid such bias in model training and improve the effectiveness of weakly supervised learning greatly.

Together, these contributions advance our understanding of efficient language model training and provide practical solutions for scenarios with limited high-quality training data. Our theoretical analyses and empirical results demonstrate the importance of carefully controlling how models learn from generated and rule-based training signals.

Introduction

The emergence of large language models has fundamentally transformed our approach to artificial intelligence and machine learning. These models have demonstrated remarkable capabilities across diverse applications, from natural language understanding to code generation, revolutionizing how we interact with and leverage computational systems. At the heart of this transformation lies a critical resource: data. Recent research has established a clear relationship between model performance and training data scale, as demonstrated by Hoffmann et al. (2022) in their comprehensive study of scaling laws. They found that the loss $L(N, D)$ of a language model with N parameters trained on D tokens can be approximated by:

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}, \tag{1}$$

where E represents the irreducible loss, and A, B, α, β are constants determined empirically. For contemporary transformer-based architectures, the computational cost in FLOPs scales approximately linearly with both model size and data: $\text{FLOPs}(N, D) \approx 6ND$. This relationship suggests that optimal model performance requires a careful balance between model size, compute resources, and training data volume.

However, this data-centric paradigm faces a significant challenge: the scarcity of high-quality language data. Recent analysis by Villalobos et al. (2022) presents a concerning projection: at current scaling rates, we may exhaust available human-generated language data by 2026. This limitation becomes even more pronounced when we consider the distribution of high-quality data across different domains. Academic papers, which represent some of our most

rigorously vetted content, contribute approximately 1 trillion tokens to the available training data. Literature, in the form of books, provides about 1.6 trillion tokens. Even in the domain of software development, where GitHub hosts vast amounts of code, the effective training data is significantly reduced after necessary preprocessing. As noted by Lozhkov et al. (2024), while GitHub contains roughly 20 trillion tokens of code, after deduplication to remove forks and filtering to eliminate non-code content such as Base64 data, only about 900 billion tokens remain suitable for training. In comparison, Meta's Llama 3 (Dubey et al., 2024) was already trained on 15 trillion tokens, a volume that significantly exceeds the total available high-quality human-generated content.

This scarcity of high-quality training data presents a fundamental challenge to the continued advancement of language models. As we approach the limits of available human-generated content, the question becomes not just how to acquire more data, but how to make better use of the data we have. This thesis addresses this challenge by exploring two complementary approaches to improving language model training without relying on scaling human-generated data: (1) leveraging model-generated and (2) leveraging rule-generated data.

We first explore an innovative pretraining framework, ELECTRA, which improves language model pretraining by leveraging model-generated corrupted data. Traditional masked language models, such as BERT, rely on randomly masking tokens and predicting them, which limits efficiency in data usage. In contrast, ELECTRA introduces a Replaced Token Detection (RTD) objective, where a generator replaces some tokens with plausible alternatives, and a discriminator learns to distinguish real from replaced tokens. This approach significantly enhances sample efficiency, as the model learns from every token rather than just the masked ones. Through theoretical analysis, we identify a critical limitation in ELECTRA's original design: inadequate control over the generator's optimization process. This lack of control can lead to suboptimal training dynamics and reduced performance. We propose a simple yet effective solution - decoupling the optimization of the generator and discriminator - which significantly improves ELECTRA's data efficiency and stability.

Our second work examines knowledge distillation, where an auxiliary ”teacher” model generates enhanced training signals to guide a ”student” model's learning. While knowledge distillation typically assumes access to additional data or a pre-trained teacher model, we demonstrate theoretically that distillation can be effective even when the teacher and student share the same training data. This means that one can always train an auxiliary teacher model to improve the the training data quality. Our analysis also reveals that the key to successful distillation lies not in the teacher's absolute performance, but in its ability to provide calibrated probability distributions that reflect underlying data patterns. Based on this understanding, we develop a novel student-oriented teacher training framework that specifically optimizes the teacher to maximize student performance rather than its own accuracy.

Our third work addresses the challenges of weakly-supervised learning, where rule-based heuristics are used to automatically generate training labels. While this approach offers a scalable way to create large labeled datasets without human annotation, we identify a fundamental issue: models trained on such data tend to inherit biases from the simple labeling rules, limiting their ability to learn more complex patterns. We propose a debiasing technique that effectively mitigates this problem by preventing models from overly relying on the specific patterns used in the labeling rules. Our method significantly improves the effectiveness of weakly-supervised learning while maintaining its scalability advantages.

Together, these works advance our understanding of how to effectively train language models in data-constrained scenarios. Our theoretical analyses provide insights into the fundamental mechanisms of learning from both model-generated and rule-generated data, while our practical solutions demonstrate substantial improvements in model performance. The methods developed in this thesis offer promising directions for continuing to advance language model capabilities even as we approach the limits of available human-generated training data.

The remainder of this thesis is organized as follows: Chapter 2 and 3 examine approaches for leveraging model-generated data, with Chapter 2 focusing on adversarial generation through ELECTRA and Chapter 3 exploring supportive generation through knowledge distillation. Chap-

ter 4 discusses our approach to learning from rule-generated data through debiased weakly-supervised learning. Finally, Chapter 5 concludes with a discussion of broader implications and future directions for efficient language model training.

Chapter 1

Model-Generated Data from "Adversarial" Models

Recent advances in language model training have demonstrated that model-generated data can serve as a powerful complement to human-generated data. While there are various ways to leverage model-generated data, one particularly promising direction is to use an adversarial model to generate training signals. This approach creates a dynamic learning environment where one model attempts to generate challenging examples while another model learns to discriminate them, potentially leading to more robust and efficient learning compared to traditional approaches.

In this chapter, we focus on ELECTRA-style pretraining, a framework that employs this adversarial learning principle. In ELECTRA, a generator model creates plausible but potentially incorrect token replacements in a text sequence, while a discriminator model learns to identify these artificial substitutions. This setup differs fundamentally from traditional masked language modeling approaches - instead of directly predicting the correct tokens, the main model learns through a detection task that can be applied to every token in the sequence, potentially improving data efficiency.

However, the success of this adversarial approach heavily depends on maintaining the right balance between the generator and discriminator. If the generator becomes too strong, it may create replacements that are too difficult to detect, hindering the discriminator's learning. Conversely, if the generator is too weak, the task may become trivial and fail to drive meaningful

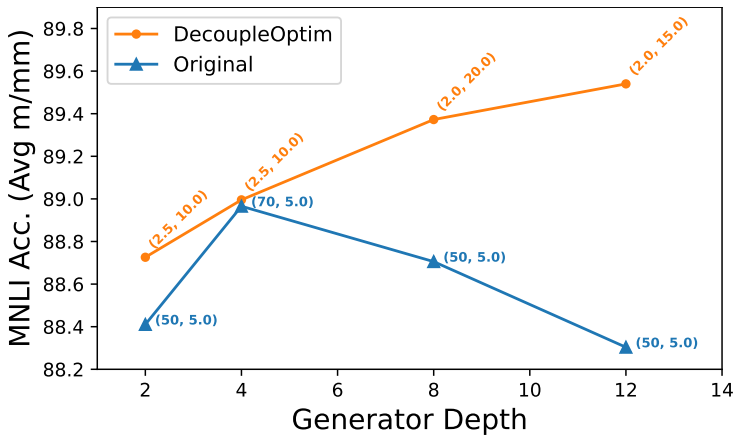


Figure 1.1. Downstream task performance (MNLI accuracy, Avg m/mm) for models trained with different generator sizes under the "Original" design and our "DecoupledOptim" technique. Each point is annotated by the best hyperparameter setting we found for this experiment, namely (loss weight, learning rate) and (generator learning rate, discriminator learning rate) for "Original" and "DecoupledOptim" respectively (learning rate is scaled by 10^4 for simplicity). More experiment details can be found in Section 1.6.1.

learning. Through our analysis, we identify critical limitations in how this balance is maintained in the original ELECTRA design and propose improvements that lead to more stable and effective training.

1.1 Introduction to Understand and Modularize Generator Optimization in ELECTRA-style Pretraining

ELECTRA-style pre-training, as introduced in Clark et al. (2020), has demonstrated significant potential in enhancing the effectiveness and efficiency of training LLMs. In specific, it trains the discriminator model (the main model that is used in downstream tasks) to detect which tokens in an input sequence were replaced by the generator model (the auxiliary model that is not used in downstream tasks). This approach has become increasingly popular among various pre-training settings and downstream applications (Clark et al., 2020; Chi et al., 2021; Kanakarajan et al., 2021; Meng et al., 2021, 2022; Bajaj et al., 2022).

Despite its effectiveness, the performance of ELECTRA is sensitive to the choice of generator size (Clark et al., 2020). As depicted in Figure 1.1, variations in generator size can lead

to a significant decline in the performance upon fine-tuning the discriminator on downstream tasks [1]. Such sensitivity inevitably demands careful selection of the generator size in real-world practices, which can be time-consuming and resource-intensive. In this research, we investigate the issue of performance degradation in a systematic manner.

First, by carefully evaluating the discriminator's capability of detecting replaced tokens, we confirm that large generator capacity can indeed hurt the effectiveness of pre-training. We also clarify that the performance degradation occurs during the pre-training stage, instead of during the fine-tuning stage. Upon further examination of the optimization in the original ELECTRA design, we recognize that it may fail to control the generator capacity effectively in the course of pre-training. The original ELECTRA relies on a weight ratio that combines the training objectives of the generator and the discriminator, in the expectation of balancing their optimization. However, this method is observed to be largely ineffective since a constant scaling of the loss will not affect adaptive optimizers like Adam (Kingma & Ba, 2015), the de facto for LLM pre-training (Liu et al., 2020b). Such a deficiency results in the sensitivity of the original ELECTRA to the generator size.

To regain control over the generator training, we modularize the generator optimization by disentangling the generator optimizer from the discriminator optimizer. This simple technique, dubbed as *DecoupledOptim*, effectively mitigate the sensitivity of ELECTRA-style pre-training to the generator size and regain the performance loss caused by a large generator. Furthermore, our algorithm can foster the flexibility of accelerating discriminator optimization without being impeded by the instability of generator training, thus bringing significant performance gain over strong baselines. We conduct experiments with the standard $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ (Devlin et al., 2019a) pre-training setting on the GLUE (Wang et al., 2018) benchmark, and our simple technique consistently outperforms the original ELECTRA design and alternative pre-training specifications that are more recently proposed.

[1] Following previous works (He et al., 2021; Bajaj et al., 2022), we use the evaluation results on MNLI (Williams et al., 2017) to indicate the performance on downstream tasks

Motivated by the empirical evidence, we turn to explore the underlying mechanism of how the generator and discriminator optimizations impact the ELECTRA-style pre-training. Our theoretical analyses reveal that a well-performed generator will indeed impair discriminator learning, which highlights the importance of controlling generator learning. Our analyses also corroborate the necessity of accelerating discriminator optimization in order to excel in pre-training performance.

To summarize, our main contributions are as follows.

- Our analysis identifies a deficiency in the original ELECTRA optimization that leads to performance degradation during the pre-training stage (Section 1.3).
- Guided by our analyses, we introduce a simple yet effective method (Section 1.4) that greatly improves training robustness and downstream performance (Section 1.6).
- We conduct theoretical analyses to further gain insights into how generator and discriminator optimizations impact the ELECTRA performance (Section 1.5).

1.2 Background and Related Work

Masked Language Modeling (MLM). MLM methods such as BERT pretrain the language model to predict randomly masked tokens in a sequence. Specifically, given an input sequence $\boldsymbol{x} = [w_1, w_2, \cdots, w_n]$, MLM generates a *masked sequence* $\tilde{\boldsymbol{x}} = [w_1, \cdots, \texttt{[mask]}, \cdots, w_n]$ by randomly selecting a few tokens at positions $\mathscr{M} = [i_1, i_2, \cdots, i_m]$ and replace them with `[mask]` token. The model is then trained to predict the original tokens given the masked sequence $\tilde{\boldsymbol{x}}$. The training objective can be formulated as

$$L(\theta) = \mathbb{E}_{\boldsymbol{x}} \sum_{i \in \mathscr{M}} -\log p_\theta(w_i | \tilde{\boldsymbol{x}})_i,$$

where θ denotes the model parameters and $p_\theta(w_i|\tilde{\boldsymbol{x}})_i$ is the predictive probability of the model at the i-th position on token w_i given the masked sequence $\tilde{x}$.

ELECTRA-style pretraining. Unlike MLM, ELECTRA constructs a pretraining task called

Replaced Token Detection (RTD) [2], which involves the joint training of two deep neural models, a generator G (auxiliary model) and a discriminator D (main model). Here the generator is pretrained with MLM as usual, while the discriminator is pretrained to detect tokens in a sequence that are replaced by a generator.

Specifically, given a masked sequence $\tilde{\boldsymbol{x}}$ constructed for MLM, a *corrupted sequence* $\hat{\boldsymbol{x}}$ is generated by replacing each `[mask]` token in $\tilde{\boldsymbol{x}}$ by a token that is sampled from the generator's predictive distribution at that `[mask]` token, namely $\hat{\boldsymbol{x}} = [w_1, \cdots, \hat{w}_i, \cdots, w_n]$ and $\hat{w}_i \sim p_G(\cdot|\tilde{\boldsymbol{x}})_i$. We refer to those sampled tokens as replaced tokens since they will be different from the original tokens at corresponding positions, as long as the generator does not predict the masked tokens correctly with a one-hot probability distribution. The discriminator is then trained to predict whether the replaced tokens in $\hat{\boldsymbol{x}}$ match the original tokens. The training objective can thus be defined as

$$L_G(\theta_D) = \mathbb{E}_{\boldsymbol{x}} \sum_{i \in \mathscr{M}} \mathbb{E}_{\hat{w}_i \sim p_G} \ell(D(\hat{\boldsymbol{x}})_i, \mathbf{1}_{\hat{w}_i = w_i}) + \mathbb{E}_{\boldsymbol{x}} \sum_{i \in [n] \setminus \mathscr{M}} \ell(D(\hat{\boldsymbol{x}})_i, 1),$$

where $D(\hat{\boldsymbol{x}})_j$ is a scalar score output by the discriminator quantifying the probability of the j-th token being replaced, ℓ is a loss function, typically binary cross-entropy (BCE), and $\mathbf{1}_{\hat{w}=w}$ is the indicator function, namely

$$\mathbf{1}_{\hat{w}=w} = \begin{cases} 1, & \text{if } \hat{w} = w, \\ 0, & \text{if } \hat{w} \neq w. \end{cases}$$

Note that in ELECTRA, the training objective of the discriminator is defined over all input tokens rather than the randomly masked subset such as that in MLM.

Analyses of ELECTRA-style pretraining. Extensive analytical efforts have been attracted to understanding the effectiveness of ELECTRA-style pretraining. It was originally believed that the RTD pretraining task gains mostly because of the improved sample efficiency by posing

[2] We will use ELECTRA-style pretraining and RTD pretraining interchangeably in this paper.

the objective on all tokens, as well as an alleviated pretraining fine-tuning gap (Clark et al., 2020). Recent works also empirically demonstrate that ELECTRA-style pretraining may be advantageous because of the low task complexity of RTD compared to MLM (Xu et al., 2020b) or implicit learning curriculum introduced by the generator (Meng et al., 2022),

Variations of ELECTRA-style pretraining. There exist various pretraining methods built on top of ELECTRA. Xu et al. (2020b) proposes a pretraining variation alike a multi-choice cloze test, where the main model predicts the original token from a small candidate set. Meng et al. (2021) introduces two additional training objectives including recovery of the original token and alignment between corrupted sequences from the same source. Hao et al. (2021) estimates the discriminator loss on replaced tokens and learns to sample difficult replace tokens from the generator. Meng et al. (2022) proceeds to automatically construct a difficult learning signal by an adversarial mixture of multiple generators. He et al. (2021) argues that embedding sharing may hurt in ELECTRA-style pretraining and suggests preventing the discriminator gradients from back-propagating to the generator embeddings. Bajaj et al. (2022) conducts a comprehensive ablation study and highlights several important improvements of ELECTRA such as large vocabulary size and relative position embedding. Zhang et al. (2022) observes the existence of "false negative" replaced tokens, namely those that are not exactly same but are synonyms to the original ones, and proposes to correct them by synonym look-up and token similarity regularization.

1.3 Impact of Generator Capacity

In ELECTRA-style pretraining, it has been widely observed that the optimal discriminator performance can only be obtained by a generator that is neither too large nor too small. As shown in Figure 1.1 for "Baseline", generators with more than 4 layers consistently hurt the discriminator performance on downstream tasks. Here, we conduct systematic analyses to explore the mechanism of this phenomenon.

1.3.1 Large Generator Capacities Slow Pre-training

Since the performance of the pre-trained model is evaluated in a two-stage setting (i.e., pre-training and fine-tuning), we first aim to understand whether the performance degradation happens in the pre-training stage (i.e., the discriminator is not trained properly) or the fine-tuning stage (i.e., the discriminator is not fine-tuned properly).

Our exploration suggests that performance degradation has already occurred during the pre-training stage. Specifically, we compare the pre-training (RTD) performance of discriminators trained with generators of different depths (4, 8, and 12 layers). As shown in Figure 1.2, discriminators trained with deeper generators achieve consistently worse RTD performance, echoing their inferior performance on downstream tasks as shown in Figure 1.1.

Note that, we used the last checkpoint of the 12-layer generator for the evaluation in Figure 1.2. Rather surprisingly, we observe that to achieve better RTD performance against a deep generator, training the discriminator on a shallow generator can be more effective than training the discriminator on that deep generator itself. As shown in Figure 1.2, the discriminator trained with the 12-layer generator performed the worst on replaced tokens sampled from this very same generator, compared to other discriminators trained with either the 4-layer or the 8-layer generator. This observation implies that the discriminator trained with a deep generator are not fully optimized in terms of their pre-training objectives and that the performance degradation may be due to a slow convergence in the course of pre-training.

It is worth mentioning that, since the relationship between the generator capacity and the performance degradation resembles overfitting, it may seem reasonable to speculate overfitting plays an important role in this phenomenon. However, in our analyses, we observe the impact of the overfitting to be marginal. We summarized our analyses on the impact of overfitting in Appendix A.4.

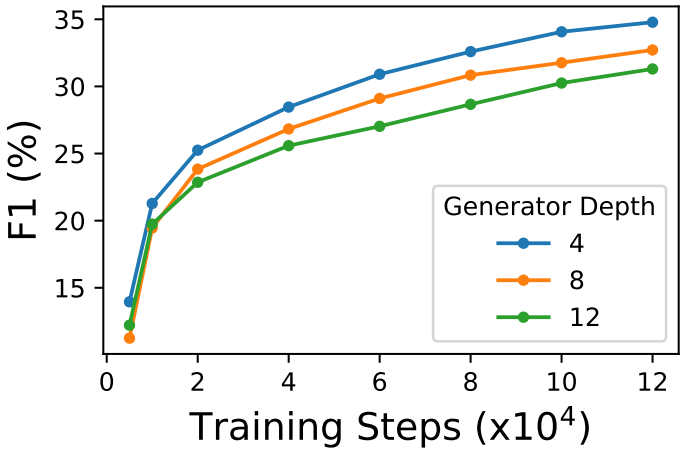


Figure 1.2. RTD performance (F1-score) of discriminators trained with generators of different depths (4, 8, 12 layers). We measure the RTD performance on replaced tokens generated by the same generator, which is the last checkpoint of the 12-layer generator used to train one of the discriminators.

1.3.2 Limitations of the Original ELECTRA in Generator Capacity Control

We can see that controlling the generator capacity is critical to the optimization of ELECTRA. Nevertheless, we found that the original design of ELECTRA may be deficient in controlling the generator capacity.

In ELECTRA, a loss weight λ is originally introduced to balance the generator optimization and discriminator optimization. In specific, the generator and the discriminator are jointly optimized through the following combined training loss (Clark et al., 2020)

$$L = L(\theta_D) + \lambda L_G(\theta_G).$$

Nevertheless, varying the value of λ may not take effect as expected. First, a constant scaling of the loss will not affect adaptive optimizers like Adam (Kingma & Ba, 2015), which is commonly used in pretraining algorithms such as ELECTRA to ensure training stability. In specific, Adam updates a model parameter by the ratio between the first moment and second

moment of its gradient, namely [3]

$$\theta := \theta - \eta \cdot \frac{\mathbb{E}[g(\theta)]}{\sqrt{\mathbb{E}[g(\theta)^2]}}, \tag{1.1}$$

where η is the learning rate, and $g(\theta) = \nabla_\theta L$ is the gradient of the model parameter θ with respect to L.

Consequently, for all generator parameters that are not shared with the discriminator (denoted as $\hat{\theta}_G$) and all discriminator parameters that are not shared with the generator ($\hat{\theta}_D$), we have [4]

$$g(\theta_G) = \nabla L(\theta_G), \quad g(\theta_D) = \lambda \cdot \nabla L_G(\theta_D).$$

It is important to note that the loss weight λ does not affect the update rule of these parameters as any constant scaling of the gradients will be canceled out in Equation (1.1). Therefore, these parameters would always be trained with the same learning rate regardless of the value of λ.

$$\theta_G := \theta_G - \eta \cdot \frac{\mathbb{E}[g(\theta_G)]}{\sqrt{\mathbb{E}[g(\theta_G)^2]}},$$

$$\theta_D := \theta_D - \eta \cdot \frac{\mathbb{E}[g(\theta_D)]}{\sqrt{\mathbb{E}[g(\theta_D)^2]}}.$$

The only parameters in ELECTRA that are affected by the loss weight λ are the embeddings θ_E shared between the generator and the discriminator. The gradients would be

$$g(\theta_E) = \nabla L(\theta_G) + \lambda \cdot \nabla L_G(\theta_D),$$

which means the update rule would become

$$\theta_E := \theta_E - \eta \cdot \frac{\mathbb{E}[\nabla L(\theta_G) + \lambda \cdot \nabla L_G(\theta_D)]}{\sqrt{\mathbb{E}[(\nabla L(\theta_G) + \lambda \cdot \nabla L_G(\theta_D))^2]}}.$$

[3] In practice, the first and second moments here are estimated as exponential moving averages.

[4] We neglect the subscript in ∇ for simplicity.

Therefore, the updates of these embeddings will be contributed by the gradients from the discriminator more if λ is larger.

Since the loss weight λ fails to balance the updates of the majority of model parameters, it cannot control the generator learning effectively. As shown in Figure 1.3, increasing the loss weight λ has little effect on the generator performance in the original ELECTRA design.

Alternative Ways to Control Generator in the original ELECTRA design. Here we discuss methods other than the loss weight λ to control the generator capacity. One way is to change the learning rate η. However, this would also alter the learning rate for discriminator learning and ultimately results in worse pretraining performance.

Another way is to reduce the model size of the generator, as also shown in Figure 1.3. This may be the only effective mechanism in the original ELECTRA design that can control generator capacity without affecting discriminator learning. However, it brings about the dependency of the pretraining performance on the careful selection of the generator size, which can be time-consuming and resource-intensive in practice.

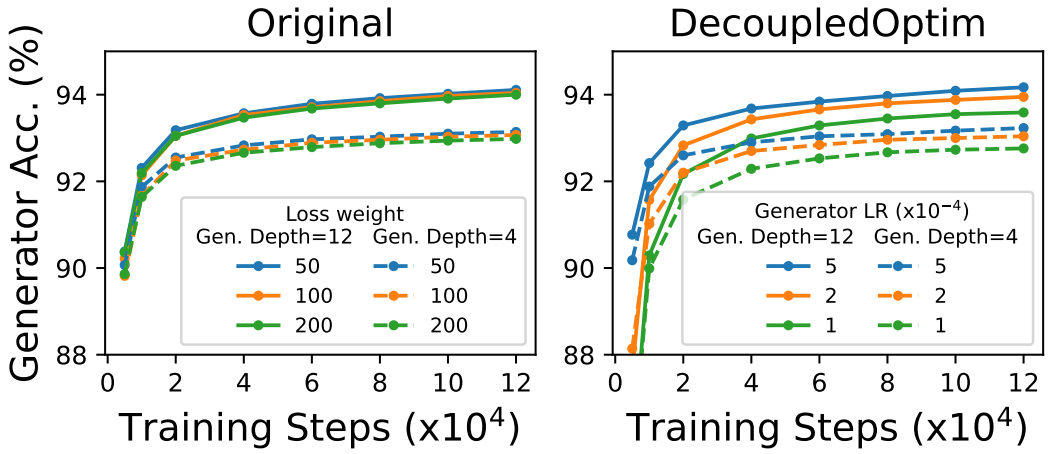


Figure 1.3. (Left): The effect of tuning the loss weight λ on the generator performance in the original ELECTRA design. (Right): The effect of tuning the generator learning rate η_G on the generator performance with our DecoupledOptim technique. For the original ELECTRA design, the learning rate is fixed as 5×10^{-4} and for our DecoupledOptim technique, the discriminator learning rate is fixed as 1×10^{-3}.

1.4 *DecoupledOptim*

Decouple the generator optimizer and the discriminator optimizer. To properly control the generator capacity, we simply decouple the generator optimizer and the discriminator optimizer. Specifically, the generator and discriminator parameters are now updated with separate rules, namely

$$\theta_G := \theta_G - \eta_G \cdot \frac{\mathbb{E}[g(\theta_G)]}{\sqrt{\mathbb{E}[g(\theta_G)^2]}},$$

$$\theta_D := \theta_D - \eta_D \cdot \frac{\mathbb{E}[g(\theta_D)]}{\sqrt{\mathbb{E}[g(\theta_D)^2]}}.$$

To control the generator capacity, we can now directly adjust the optimizer dedicated for the generator (*e.g.*, adjusting η_G, the generator learning rate). Figure 1.3 shows that, for a large generator, we can simply reduce η_G to effectively control its capacity during the pre-training.

This implies that DecoupledOptim is capable to handle large generators and reduce the sensitivity of ELECTRA-style pretraining on the choice of the generator size. As elaborated in Section 1.6, our proposed method is simple yet effective, consistently outperforming the original ELECTRA design and its recently proposed variants.

Note that for the simplicity of the implementation, we will not share the embeddings between the generator and the discriminator anymore. Despite the belief that embedding sharing is crucial in ELECTRA-style pretraining since RTD pretraining may not be as effective as MLM in learning token representations (He et al., 2021), we found that with our decoupled optimization, a discriminator learned from randomly initialized embeddings can in fact achieve equivalently good or even better performance.

Improve ELECTRA with Sufficient Discriminator Optimization. With our decoupled-optimizer design, we can in fact not only control generator capacity more easily but also achieve significantly better pretraining performance by optimizing the discriminator more sufficiently.

In the original ELECTRA design, the same learning rate is assigned to the generator and discriminator as mentioned in Section 1.3.2. Therefore, attempts to speed up discriminator

optimization by increasing the learning rate inevitably lead to larger generator capacity, thus yielding worse pretraining performance. Moreover, The increased learning rate may even cause training failure since the generator training in an MLM style can undergo strong instability with a large learning rate. In our experiments, we found that the original ELECTRA design diverges within 25K training steps, despite a proper selection of the loss weight (*e.g.*, 50) and the generator size (*e.g.*, 4 layers).

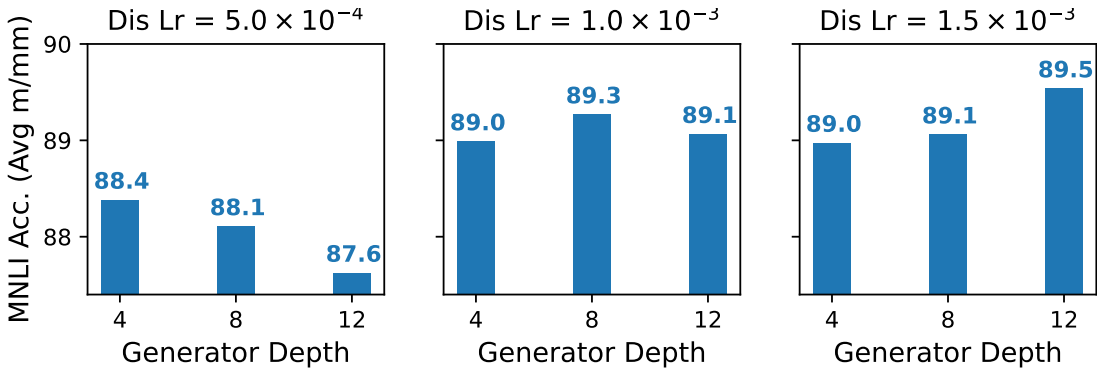


Figure 1.4. Discriminator performance on downstream tasks (MNLI) with different generator sizes and discriminator learning rates. Here the generator learning rate is fixed as 2×10^{-4}.

However, with the optimizers decoupled in DecoupledOptim, we can now accelerate discriminator optimization without being impeded by generator learning. Empirical experiments in Figure 1.4 show that, as we increase the discriminator learning rate such that its optimization becomes more sufficient, the pretraining performance is increasingly good. One may notice that the discriminator learning rate can often be as large as 1.5×10^{-3}, which is 3-7 times the learning rate suitable for generator training. Furthermore, we observe that with increasingly sufficient discriminator optimization, the best generator shifts to one with a larger capacity, even as large as the discriminator itself (12 layers).

1.5 Optimization of ELECTRA-style Methods

Here, we conduct theoretical analyses to gain insight into how generator and discriminator optimizations impact the performance of the ELECTRA-style pretraining.

Problem setup. We consider a simplified RTD task where only one token in an input sequence is replaced. We refer to the rest of those unchanged tokens in this sequence as context. Let w be a word in the sentence, and let c be the remaining context words in the same sentence. The generator is trained to predict the original token given the context, namely

$$L(\theta_G) = \mathbb{E}_{c,w} - \log p_G(w|c). \tag{1.2}$$

For discriminator training, we focus on the detection of this single replaced token exclusively. The optimization objective of the discriminator D can be thus described as

$$\bar{L}_G(\theta_D) = \mathbb{E}_{c,w}\mathbb{E}_{\hat{w}\sim p_G}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w}). \tag{1.3}$$

Ideal discriminator optimization objective. An ideal discriminator optimization objective should align with the discriminator performance on downstream tasks. However, discriminator performance evaluated against a given replaced token distribution may not always be indicative of the downstream performance (see more in Section A.4). Ideally, a discriminator should be able to detect any possible tokens replaced in a sequence, regardless of the specific distribution from which such replaced tokens are sampled. To this end, we define the ideal optimization objective of the discriminator as the highest possible discriminator loss achieved by any replaced token distribution, namely the probability distributions from which the replaced tokens are sampled.

Definition 1 (Ideal optimization objective of the discriminator)**.** *Let $\mathscr{P}$ be a family of replaced token distributions. The ideal optimization objective of the discriminator D can be defined as*

$$L^*(\theta_D) = \mathbb{E}_{c,w} \sup_{p\in\mathscr{P}} \mathbb{E}_{\hat{w}\sim p}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w}). \tag{1.4}$$

Practical optimization of the discriminator. In practice, such an ideal optimization objective cannot be used as a loss function for training the discriminator since it is not feasible to enumerate

all possible replaced token distributions. Fortunately, if we have an assumption on the "difficulty" of the replaced token distributions, we can bound the highest discriminator loss over all possible replaced token distributions. This upper bound can further be approached by the discriminator loss on the generator's predictive distribution as a replaced token distribution. Hence, the optimization objective defined by a generator (*i.e.*, Equation (1.3)) can be a fair surrogate of the ideal objective.

Assumption 1 (Difficulty of the replaced token distribution)**.** *We assume a discriminator is more likely to make detection errors if the sampled replaced tokens recover the original token more frequently, namely*

$$\mathbb{E}_{\hat{w}\sim p}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w}) = F_D(\mathbb{E}_{\hat{w}\sim p}[\mathbf{1}_{\hat{w}=w}]) \tag{1.5}$$

where $F_D : [0,1] \to \mathbb{R}^+$ is a concave and monotonically increasing function that is dependent on the discriminator D.

We can now show that the ideal discriminator objective is bounded by the surrogate objective defined by a generator. The intuition here is that, given Assumption 1, the predictive distribution of the generator should approximate the most difficult replaced token distribution, and the approximate error happens to be bounded by the performance of the generator.

Lemma 1 (The discriminator objective defined by a generator is a surrogate of the ideal objective)**.** *Let $m_D = \max_{c,w} \|\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w})\|_2$ be the upper bound on the discriminator loss given any context-token pairs. Then we have*

$$L^*(\theta_D) \leq \bar{L}_G(\theta_D) + 2^{-1/2} m_D L(\theta_G)^{1/2}. \tag{1.6}$$

Lemma 1 implies that training towards a surrogate objective defined by a generator $L_G(\theta_D)$ can indeed optimize the ideal discriminator objective. This justifies the basic ELECTRA design which employs a generator to sample replaced tokens for discriminator training. It also

implies that an under-performed generator may not be as effective for optimizing the ideal objective since the distribution approximation error (the 2-nd term) would be much higher.

Large generator capacity may hurt optimization. However, well-performed generators may be less effective for optimizing the ideal objective as well. This is because well-performed generators will approach the most difficult replace token distribution based on Assumption 1, thus creating significantly higher discriminator loss $L_G(\theta_D)$ in Equation (1.6). We have the following Lemma to demonstrate this.

Lemma 2 (Dependence of the discriminator loss on generator performance)**.**
Let $V_G = \mathbb{E}_{c,w}[(-\log p_G(w|c) - L(\theta_G))^2]$ *be the variance of the generator loss, we have*

$$\bar{L}_G(\theta_D) \leq F_D\left((1+V_G/2)\, e^{-\bar{L}(\theta_G)} + V_G/2\; e^{\varepsilon}\right). \tag{1.7}$$

Lemma 2 shows that the discriminator loss given a generator is inversely correlated with the generator loss. This means strong generators may create significantly higher discriminator loss. If such a high discriminator loss cannot be sufficiently reduced through the optimization process, which is likely since the training budget is always limited, the ideal objective cannot be sufficiently optimized as well.

Modeling the generator and discriminator optimization. To further illustrate the effects of both the generator and discriminator optimizations, we introduce a simplified modeling of the optimization process, which is based on trajectory analysis of gradient descent for deep linear neural networks (Arora et al., 2018).

Proposition 1 (Gradient descent trajectory of deep linear neural networks (informal))**.** *In gradient descent, let* $\theta(t)$ *be the model parameters after t updates,* η *be the learning rate that meets certain regularities, and N be the number of layers in the model, then we have*

$$\mathscr{L}(\theta(t)) \leq \mathscr{L}(\theta(0)) \cdot (1 - \eta \cdot c^{\frac{2(N-1)}{N}})^t, \tag{1.8}$$

where c is a positive constant.

We can now derive a complete picture of the effect of the generator performance on the optimization of the discriminator. Together with Lemmas 1 and 2, we have the following result.

Theorem 1 (Optimization of the ideal discriminator objective)**.** *Consider the discriminator optimization after the generator is trained with several updates. Let η_G be the generator's learning rate and N_G be the generator depth. The generator loss after t_G updates is*

$$L(\theta_G(t_G)) = L(\theta_G(0))\,(1-\eta_G\cdot\xi_G)^{t_G}\,, \tag{1.9}$$

where $\xi_G = c^{2(N_G-1)/N_G}$. Subsequently, let η_D be the discriminator's learning rate and N_D be the discriminator depth. Then after t_D discriminator updates, we have

$$\begin{aligned} L^*(\theta_D(t_D)) \leq & F\left(e^{-L(\theta_G(t_G))}\right)\cdot(1-\eta_D\cdot\xi_D)^{t_D} \\ & + 2^{-1/2} m_D\,[L(\theta_G(t_G))]^{1/2}\,, \end{aligned} \tag{1.10}$$

where we have neglected some constants in Equation (1.7) for simplicity.

One can see that, in terms of generator optimization, increasing the generator learning rate first helps and then hurts the discriminator performance for a given discriminator setting, as also illustrated in Figure 1.5. This results in a sweet spot where the generator learning rate is the best for the discriminator performance. Similarly, since $\xi \propto N$, increasing the generator depth would also first help and then hurt the discriminator performance.

In terms of discriminator optimization, increasing the discriminator learning rate or depth can almost always help the discriminator performance. In an ideal case, if the discriminator optimization is sufficient, for example, by letting $\eta_D\cdot\xi_D \approx 1$ or $t_D \approx \infty$ in Equation (1.10), then the discriminator performance will improve monotonically with the generator performance, as also illustrated in Figure 1.5 where the discriminator learning rate is sufficiently large. Unfortunately,

it is not possible to always sufficiently optimized the discriminator in practice given training instability and/or limited training budgets.

Interestingly, Equation (1.10) shows that, with increasingly sufficient discriminator optimization, the best generator should shift to one with a larger capacity, as also illustrated in Figure 1.5. One may find this echoes the empirical observation in Figure 1.4.

Finally, Equation (1.10) also reflects the limitation of the original ELECTRA design mentioned in Section 1.3.2. As also illustrated in Figure 1.5, since the generator and discriminator are assigned almost the same learning rate, the original design can only reach a line (1-D subspace) in the entire optimization space. Further, due to the training instability of the generator, this line is truncated where the discriminator learning rate is still small and thus the discriminator performance is still suboptimal. In contrast, DecoupledOptim can increase the discriminator learning rate without being affected by the generator learning thus achieving better performance by exploring the entire optimization space.

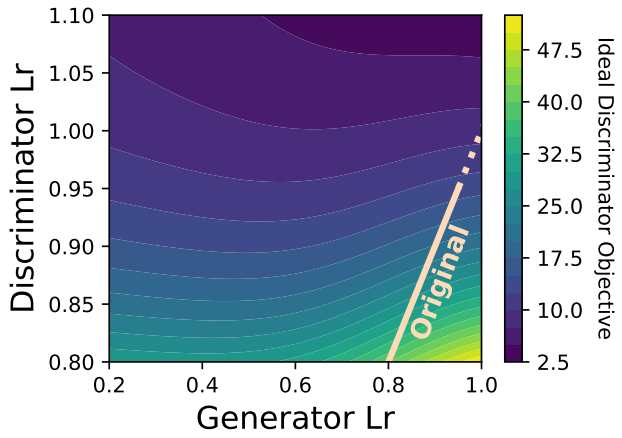


Figure 1.5. Illustration of the discriminator performance in terms of the ideal objective (denoted by the color, and the darker color corresponds to better performance) with respect to the generator learning rate (x-axis) and discriminator learning rate (y-axis). The line represents the space that the original ELECTRA design can possibly reach by modulating its learning rate in an ideal scenario. The dashed line represents the space that the original ELECTRA fails to reach in practice due to the training instability of the generator.

Table 1.1. Results on the GLUE development set. "†" indicates the model is pretrained for 1M updates with batch size of 256. "‡" indicates the model is pretrained for 100K updates with batch size of 8K. "-" indicates that no public reports are available.

Model	MNLI-(m/mm) (Acc.)	QQP (Acc.)	QNLI (Acc.)	SST-2 (Acc.)	CoLA (Mat. Corr.)	RTE (Acc.)	MRPC (Acc.)	STS-B (Spear. Corr.)	AVG
Base Setting									
BERT (Devlin et al., 2019a)	84.5/ -	91.3	91.7	93.2	58.9	68.6	87.3	89.5	83.1
RoBERTa (Liu et al., 2019b)	85.8/85.5	91.3	92.0	93.7	60.1	68.2	87.3	88.5	83.3
XLNet (Yang et al., 2019)	85.8/85.4	-	-	92.7	-	-	-	-	-
DeBERTa (He et al., 2020)	86.3/86.2	-	-	-	-	-	-	-	-
TUPE (Ke et al., 2020)	86.2/86.2	91.3	92.2	93.3	63.6	73.6	89.9	89.2	84.9
ELECTRA (Clark et al., 2020)	86.9/86.7	91.9	92.6	93.6	66.2	75.1	88.2	89.7	85.5
MC-BERT (Xu et al., 2020b)	85.7/85.2	89.7	91.3	92.3	62.1	75.0	86.0	88.0	83.7
COCO-LM (Meng et al., 2021)	88.5/88.3	92.0	93.1	93.2	63.9	84.8	91.4	90.3	87.2
AMOS (Meng et al., 2022)	88.9/88.7	92.3	93.6	94.2	**70.7**	86.6	90.9	**91.6**	88.6
DeBERTaV3 (He et al., 2021)	89.3/89.0	-	-	-	-	-	-	-	-
METRO (Bajaj et al., 2022)	89.0/88.8	92.2	93.4	**95.0**	70.6	86.5	91.2	91.2	88.6
$\text{METRO}_{\text{ReImp}}$	89.0/88.9	92.0	93.4	94.4	70.1	86.3	91.4	91.2	88.5
DecoupledOptim	**89.4/89.7**	**92.4**	**93.6**	94.7	70.6	**88.8**	**92.2**	91.1	**89.1**
Large Setting									
BERT†	86.6/ -	-	-	-	-	-	-	-	-
RoBERTa‡	89.0/ -	91.9	93.9	95.3	66.3	84.5	90.2	91.6	87.8
XLNet†	88.4/ -	91.8	93.9	94.4	65.2	81.2	90.0	91.1	87.0
TUPE†	88.2/88.2	91.7	93.6	95.0	67.5	81.7	90.1	90.7	87.3
$\text{METRO}_{\text{ReImp}}$	89.9/90.2	**92.5**	94.5	94.3	69.7	**88.8**	**91.9**	91.6	89.2
DecoupledOptim	**90.5/90.6**	92.4	**94.7**	**96.1**	**72.1**	88.4	91.2	**92.2**	**89.7**

1.6 Experiments

1.6.1 Experiment Setup

Pretraining Setup. We conduct experiments with two standard settings, *Base* and *Large*, following previous works (Devlin et al., 2019a; Meng et al., 2021; Bajaj et al., 2022). Specifically, we employ Wikipedia and BookCorpus (Zhu et al., 2015) (16 GB of texts, 256M samples) for pretraining with sequence length as 512. We use a cased sentence piece BPE vocabulary of 128K tokens following (He et al., 2020), since larger vocabulary size improves LLMs without significant additional training and inference cost (Bao et al., 2020).

We conduct pretraining for 125K updates with a batch size of 2048. For our DecoupledOptim, we use the same hyperparameter combination in both Base and Large settings, namely the generator learning rate is set as 2×10^{-4} and the discriminator learning rate is set as 1.5×10^{-3}. Detailed hyperparameter settings can be found in Appendix A.2.

Model Architecture. Our main model (discriminator) in the Base setting follows the $\text{BERT}_{\text{base}}$ architecture (Devlin et al., 2019a), namely a 12-layer transformer with 768 hidden dimensions plus T5 relative position encoding (Raffel et al., 2019) with 32 bins. We employ Admin (Liu et al., 2020b, 2021) for model initialization to stabilize the training. Our main model in the Large setting follows $\text{BERT}_{\text{Large}}$, namely a 24-layer transformer with 1024 hidden dimensions and 128 relative position encoding bins. Our auxiliary model (generator) in Base has the same architecture as the main model, which is larger than the recommended size in previous works (Clark et al., 2020; Meng et al., 2021) (typically 4 layers), but yields significantly better results. Our auxiliary model in Large has 8 layers with other settings same as the main model.

Downstream evaluation setup. We conduct the evaluation on downstream tasks following the setup in previous works (Meng et al., 2021; Bajaj et al., 2022). Specifically, we evaluate on GLUE (Wang et al., 2018) language understanding benchmark with a single-task, single-model fine-tuning setting following previous works. We employ the suggested training hyperparameters such as the AdaMax optimizer (Kingma & Ba, 2015) from Liu et al. (2019a, 2020c). We report Spearman correlation on STS-B, Matthews correlation on CoLA, and accuracy on the rest of the datasets. Detailed hyperparameter settings can be found in Appendix A.2.

Baselines. We compare with various pretrained models that are consistent with our basic settings (dataset and training steps) (see Table 1.1). For the Large setting, we also compare with pretrained models that consume similar computation costs, for example, in terms of the total number of processed tokens. We report the results of baseline models from the corresponding papers and their follow-up works, whichever are higher. We also reimplement METRO as our baselines [5].

[5] Both the re-implemented METRO and our method are implemented within the same codebase, which is built on top of FAIRSEQ, a popular open-sourced package (Ott et al., 2019).

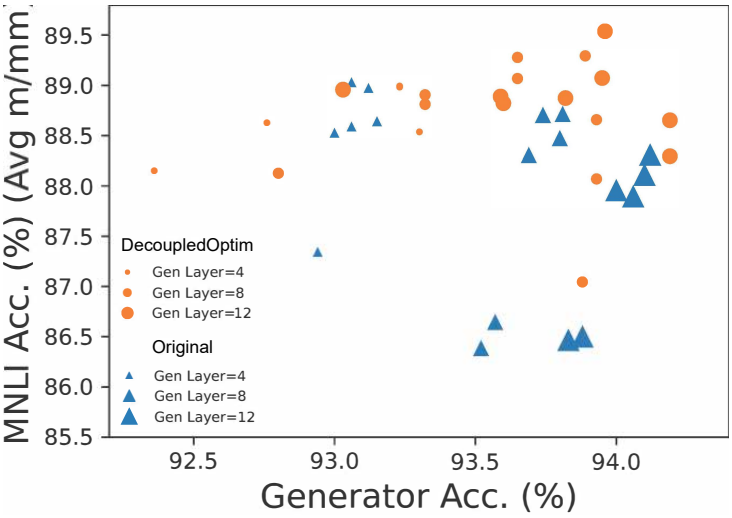


Figure 1.6. Discriminator performance on downstream tasks (MNLI Avg m/mm) versus the generator performance for a variety of hyperparameter combinations that can affect the generator training. For the original ELECTRA design (denoted by "▲") specifically, we modulate the loss weight λ in $\{50, 70, 100, 200\}$ and the learning rate in $\{10, 5, 2, 1\} \times 10^{-4}$. For our DecoupledOptim technique (denoted by "•") specifically, we modulate the generator learning rate also in $\{10, 5, 2, 1\} \times 10^{-4}$. For both, we modulate the generator depth in $\{4, 8, 12\}$. For diverged training, the combination of hyperparameters is not included in the figure.

1.6.2 Results

Main Results. Table 1.1 lists the downstream evaluation results of DecoupledOptim and competitive baselines under the Base and Large setting. DecoupledOptim outperforms previous state-of-the-art by notable margins in terms of both the overall GLUE score and specific results on large datasets, which are considered to be more reliable.

Robustness to Generator Setting. We experiment with a wide variety of pretraining hyperparameters in the Base setting to validate the robustness of DecoupledOptim with respect to the change of generator capacity, since it is the main focus of this paper. We use our reimplemented METRO as a strong baseline. As shown in Figure 1.6, DecoupledOptim yields more stable downstream performance when the generator capacity varies/the hyperparameter changes. DecoupledOptim also achieves consistently better performance when employing generators with larger sizes.

Pretraining Efficiency. In each pretraining step, DecoupledOptim introduces no additional

model parameters or computation cost compared to the original ELECTRA design. Note that separate generator and discriminator embeddings in DecoupledOptim require exactly the same amount of operations as the shared embeddings in the original ELECTRA design, since the gradients of the embeddings have to be back-propagated from the generator and discriminator loss separately in both scenarios. DecoupledOptim does induce additional memory consumption due to separate embeddings. To mitigate this, one can maintain the shared embedding while employing separate optimizers, by merging the embedding gradients in a custom manner, which we would like to leave as a future work.

1.7 Summary

In this chapter, we conduct systematic analyses on the impact of generator capacity on the performance of the discriminator in ELECTRA-style pretraining. Our investigation begins with the observation that using a large auxiliary generator often results in a degradation of the downstream performance of the main discriminator model. Our analyses suggest that such performance degeneration is due to inadequate control of the generator capacity during pretraining, highlighting a long-overlooked issue in ELECTRA-style training. Based on our findings, we propose a simple-yet-effective method that greatly improves the training robustness and downstream performance, which further verified our intuition.

This chapter incorporates material from the publication "Understand and Modularize Generator Optimization in ELECTRA-style Pretraining" by Chengyu Dong, Liyuan Liu, Hao Cheng, Jingbo Shang, Jianfeng Gao, Xiaodong Liu, published in Proceedings of the 40th International Conference on Machine Learning (ICML 2023). The dissertation author was the primary investigator and lead author of this paper.

Chapter 2

Model-Generated Data from "Supportive" Models

The previous chapter explored how model-generated data can enhance training through an adversarial model, where a generator model creates challenging examples to drive the main model's learning. While this adversarial approach proves effective, it represents just one way of leveraging model-generated data. An alternative and complementary approach is to use model-generated data in a supportive rather than adversarial manner, where an auxiliary model actively guides the main model's learning by providing enhanced training signals.

Knowledge distillation exemplifies this supportive approach to using model-generated data. Instead of creating challenging examples, a "teacher" model generates probability distributions over possible outputs that can help guide a "student" model's learning. These soft probability distributions often capture nuanced relationships in the data that are not apparent from the hard labels alone. For example, when classifying images of numbers, a teacher might indicate that a particular "7" also shares some visual features with "1", providing richer learning signals than the simple label "7".

However, similar to the optimization challenges we encountered with ELECTRA's adversarial generator, the effectiveness of knowledge distillation heavily depends on how we train and utilize the supportive teacher model. While conventional wisdom suggests that a better-performing teacher leads to better student learning, our analysis reveals a more nuanced

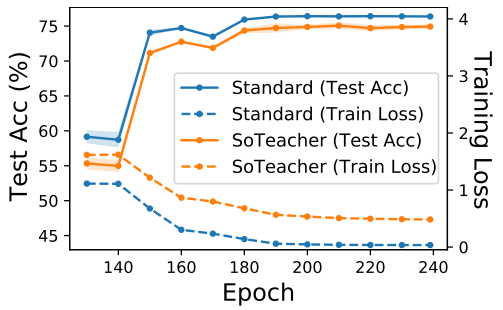
relationship. The key to successful distillation may not lie in the teacher's absolute performance, but rather in how well it generates training signals that are specifically beneficial for student learning. In this chapter, we conduct a thorough theoretical and empirical investigation of knowledge distillation, focusing particularly on how to optimize the teacher model to generate the most effective training signals for the student.

2.1 Introduction to Toward Student-Oriented Teacher Network Training For Knowledge Distillation

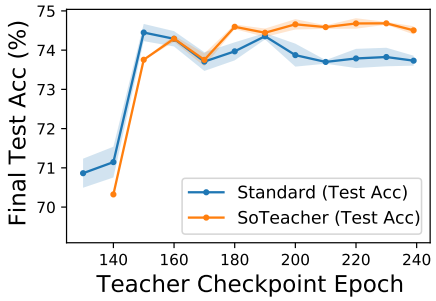
Knowledge distillation aims to train a small yet effective *student* neural network following the guidance of a large *teacher* neural network (Hinton et al., 2015). It dates back to the pioneering idea of model compression (Buciluǎ et al., 2006) and has a wide spectrum of real-world applications, such as recommender systems (Tang & Wang, 2018; Zhang et al., 2020), question answering systems (Yang et al., 2020; Wang et al., 2020) and machine translation (Liu et al., 2020a).

Despite the prosperous interests in knowledge distillation, one of its crucial components, teacher training, is largely neglected. The existing practice of teacher training is often directly targeted at maximizing the performance of the teacher, which does not necessarily transfer to the performance of the student. Empirical evidence shows that a teacher trained toward convergence will yield an inferior student (Cho & Hariharan, 2019) and regularization methods benefitting the teacher may contradictorily degrade student performance (Müller et al., 2019). As also shown in Figure 2.1, the teacher trained toward convergence will consistently reduce the performance of the student after a certain point. This suggests a fundamental discrepancy between the common practice in teacher training and the ideal learning objective of the teacher that orients toward student performance.

In this work, we explore both the theoretical feasibility and practical methodology of training the teacher toward student performance. Our analyses are built upon the recent



(a) Teacher performance with different numbers of teacher training steps.



(b) Student performance when distilled from different teacher checkpoints.

Figure 2.1. We train teacher models on CIFAR-100, saving a checkpoint every 10 epochs, and then use this checkpoint to train a student model through knowledge distillation. With our method, the teacher is trained with a focus on improving student performance, leading to better student performance even if the teacher's own performance is not as high.

understanding of knowledge distillation from a statistical perspective. In specific, Menon et al. (2021) show that the soft prediction provided by the teacher is essentially an approximation to the true label distribution, and true label distribution as supervision for the student improves the generalization bound compared to one-hot labels. Dao et al. (2021) show that the accuracy of the student is directly bounded by the distance between teacher's prediction and the true label distribution through the Rademacher analysis.

Based on the above understanding, a teacher benefitting the student should be able to learn the true label distribution of the *distillation data* [1]. Since practically the distillation data is often reused from the teacher's training data, the teacher will have to learn the true label distribution of its own training data. This might appear to be infeasible using standard empirical risk minimization, as the teacher network often has enough capacity to fit all one-hot training labels, in which case, distilling from teacher predictions should largely degrade to directly training with one-hot labels. Existing works tend to evade this dilemma by distilling from teacher predictions only on data that is not used in teacher training (Menon et al., 2021; Dao et al., 2021).

Instead, we directly prove the feasibility of training the teacher to learn the true label

[1] For simplicity, we refer to the training data of the student model in knowledge distillation as the distillation data (Stanton et al., 2021)

distribution of its training data. We show that the standard empirical risk minimizer can approach the true label distribution of training data under a mixed-feature data distribution, as long as the feature extractor of the learner network is Lipschitz continuous and is robust to feature transformations.

In light of our theory, we show that explicitly imposing the Lipschitz and consistency constraint in teacher training can facilitate the learning of the true label distribution and thus improve the student performance. We conduct extensive experiments on two benchmark datasets using various knowledge distillation algorithms and different teacher-student architecture pairs. The results confirm that our method can improve student performance consistently and significantly.

We believe our work is among the first attempts to explore the theory and practice of training a *student-oriented* teacher in knowledge distillation. To summarize, our main contributions are as follows.

- We show that it is theoretically feasible to train the teacher to learn the true label distribution of the distillation data even with data reuse, explaining the effectiveness of the current knowledge distillation practice.
- We show that by adding Lipschitz and consistency regularization during teacher training, it can better learn the true label distribution and improve knowledge consistently.

2.2 Preliminaries

We study knowledge distillation in the context of multi-class classification. Specifically, we are given a set of training samples $\mathscr{D} = \{(x^{(i)}, y^{(i)})\}_{i\in[N]}$, where $[N] \coloneqq \{1, 2, \cdots, N\}$. $\mathscr{D}$ is drawn from a probability distribution $p_{X,Y}$ that is defined jointly over input space $\mathscr{X}$ and label space $\mathscr{Y} = [K]$. For convenience, we denote $1(y) \in \mathbb{R}^K$ as the one-hot encoding of label y.

Learning Objective of Teacher in Practice. In common practice, the teacher network f is

trained to minimize the empirical risk given a loss function ℓ, namely

$$\min_f \mathbb{E}_{(x,y)\in\mathscr{D}}\, \ell(f(x),y), \tag{2.1}$$

where $\mathbb{E}_{(x,y)\in\mathscr{D}}$ is the empirical expectation.

Ideal Learning Objective of Teacher. Recent advances in understanding knowledge distillation suggest a teacher should approximate the true label distribution of the distillation data, which is often reused from the teacher's training data. From this perspective, ideally the teacher should learn the true label distribution of its training data, namely

$$\min_f \mathbb{E}_{(x,y)\in\mathscr{D}}\, \|f(x) - p^*(x)\|. \tag{2.2}$$

Here, $p^*(x) \coloneqq p_{Y|X}(\cdot|x)$ denotes the *true label distribution* of an input x, namely the (unknown) category distribution that its label is sampled from, which is not necessarily one-hot. And $\|\cdot\|$ can be an arbitrary p-norm.

Our Research Questions. One can find that there is a fundamental discrepancy between the learning objective of the teacher in the common practice and the ideal learning objective of the teacher. In particular, minimization of Eq. (2.1) would lead to $f(x) = 1(y)$ for any input x, which significantly deviates from $p^*(x)$ and thus challenges the effectiveness of knowledge distillation. Therefore, in this work, we explore the following two questions.

(i) *Can a teacher network learn the true label distribution of the training data with the standard teacher training practice?*

(ii) *How to train a teacher to better learn the true label distribution and improve student performance?*

We will present our findings of these two questions in Sec. 2.3 and Sec. 2.4, respectively.

2.3 Theoretical Feasibility to Learn True Label Distribution of Training Data

We now explore the theoretical feasibility of training a teacher network that learns the true label distribution of the training data under the empirical risk minimization framework. Unless specifically stated otherwise, by "data" we refer to the data used to train the teacher, and by "network" we refer to the teacher network. Note that throughout the discussion here, our major focus is the *existence* of a proper minimizer, instead of the details of the optimization process.

2.3.1 Notations and Problem Setup

Notation. Here, we introduce our notations in addition to the ones mentioned in Section 2.2. We will use calligraphic typefaces to denote sets, *e.g.*, the dataset $\mathscr{D}$. We use $|\mathscr{D}|$ to denote the size of the set. We use $\circ$ to denote function composition. We use $\tilde{O}(\eta)$ to denote polynomial terms of η.

Data Distribution. We consider a data distribution which we refer to as the *mixed-feature distribution*. Our distribution can be viewed as a simplified version of the "multi-view distribution" introduced in Allen-Zhu & Li (2020). Our distribution can also be viewed as a variation of Latent Dirichlet Allocation (LDA) (Blei et al., 2001), a generative data distribution widely used to model text data.

Definition 2 (Mixed-feature distribution)**.** *We first generate the input x following LDA. We define a feature vocabulary $\mathscr{Z}$ that denotes the names of all possible features in the data inputs. For example, in image classification we can have $\mathscr{Z} = \{$'eye', 'tail', 'wheel', ...$\}$. Now for each data example i,*

1. *Sample M feature names from $\mathscr{Z}$, namely $z_m \sim p_Z$, where p_Z is a discrete distribution defined over $\mathscr{Z}$.*

2. *For each feature name z, we sample its input representation $x_z \in \mathbb{R}^b$, namely $x_z \sim p_X(\cdot|z)$, where $p_X(\cdot|z)$ is a continuous distribution with finite variance, namely $Var[X|z] \leq \nu_z$. The finite variance here means that the representation of the same feature sampled in different inputs should be similar.*

3. *For each feature name z, we transform its input representation by a function $\gamma : \mathbb{R}^b \to \mathbb{R}^b$, where γ is sampled from a set of possible transformation functions $\mathscr{T}$, namely $x_z := \gamma(x_z)$. For example, in image classification γ can be rotation, mirroring, or resizing.*

4. *We concatenate the transformed representations of these features to obtain the input, namely $x = (x_{z_1}, x_{z_2}, \cdots, x_{z_M})$, where $x \in \mathbb{R}^{b \times M}$. This views each data input as a concatenation of M patches and each patch is a b-dimensional vector, which follows the assumption in Allen-Zhu & Li (2020).*

Next, we generate the label y. We assume each feature name defines a label distribution $p_Y(\cdot|z)$. The label distribution of an input x is the geometric average of the label distributions of the feature names, namely $y \sim p_Y(\cdot|x) := (\prod_m p_Y(\cdot|z_m))^{1/M}$.

Note that in our data distribution, the assumption that each patch is a vector is solely for the simplicity of illustration and can be trivially generalized, as our theory is not dependent on it. The specific shape of each patch can be arbitrary, for example, can be a 3-rank tensor (height, width, channel) in image classification.

One difference between our data distribution and "multi-view distribution" is that one can define different label sets $\mathscr{Y}$ for the same input data under our data distribution. This is more realistic as it models datasets that have coarse-grained/fine-grained label sets respectively but the same input data.

Network Architecture. We consider a multi-layer neural network that produces probabilistic outputs, namely $f : \mathbb{R}^{b \times M} \to [0,1]^K$. The network consists of a feature extractor and a classification head, namely $f := f_C \circ f_E$, defined as follows respectively.

- $f_E : \mathbb{R}^{b\times M} \to \mathbb{R}^{M\times d}$ denotes the feature extractor, whose architecture can be arbitrary as long as it processes each patch independently. With a little abuse of notation, we will also write $h_m \coloneqq f_E(x_m)$.

- $f_C : \mathbb{R}^{M\times d} \to \mathbb{R}^K$ denotes the probabilistic classification head, which consists of a 1x1 convolutional layer and a modified Softmax layer, namely $f_C(h) = \widetilde{\text{Softmax}}(w_C h)$, where $w_C \in \mathbb{R}^{1\times K\times d}$. The 1x1 convolutional layer is similar to the assumption in (Allen-Zhu & Li, 2020), while the modified Softmax is slightly different from the standard Softmax in terms of the denominator, namely $\widetilde{\text{Softmax}}(\hat{h}) = \frac{\exp(1/M\sum_m \hat{h}_m)}{(\prod_m \sum_k \exp(\hat{h}_{m,k}))^{1/M}}$, where $\hat{h} \coloneqq w_c h$.

2.3.2 A Hypothetical Case: Invariant Feature Extractor

For starters, we investigate a hypothetical case where the feature extractor is *invariant*, which means that it can always produce the same feature map given the same feature patch, regardless of which input this feature patch is located at or which transformation is applied to this feature patch. Given such an assumption, showing the learning of true label distribution of the training data can be greatly simplified. We will thus briefly walk through the steps towards this goal to shed some intuitions.

Definition 3 (Invariant feature extractor)**.** *We call f_E an invariant feature extractor, if for any two inputs i and j, for any two transformations γ and γ', $f_E(\gamma(x_m^{(i)})) = f_E(\gamma'(x_{m'}^{(j)}))$, as long as $z_m = z_{m'}$, namely the feature names of the patches m and m' match.*

We first show that given an invariant feature extractor, the minimizer of the empirical risk has the property that its probabilistic prediction of each feature converges to the sample mean of the labels whose corresponding inputs contain this feature.

Lemma 3 (Convergence of the probabilistic predictions of features)**.**
Let $\bar{y}_z \coloneqq \frac{1}{N}\sum_{\{i|z\in\mathscr{Z}^{(i)}\}} 1(y^{(i)})$, where $\mathscr{Z}^{(i)}$ denote the set of feature names in the i-th input, and thus $\{i|z \in \mathscr{Z}^{(i)}\}$ denotes the set of inputs that contain feature z. Let f^* be a minimizer of

the empirical risk (Eq. (2.1)) and assume f_E^* is an invariant feature extractor. Let $p_{f^*}(x_z) \coloneqq Softmax(w_C f_E^*(x_z))$ be the probabilistic prediction of feature z. We have

$$p_{f^*}(x_z) = \bar{y}_z. \tag{2.3}$$

The intuition here is that since the feature extractor is invariant and the number of possible features is limited, the feature maps fed to the classification head would be a concatenation of M vectors selected from a fixed set of $|\mathscr{Z}|$ candidate vectors, where each vector corresponds to one feature in the vocabulary. Therefore, the empirical risk can be regrouped as $-\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log f^*(x^{(i)}) = -\frac{1}{M}\sum_{z\in\mathscr{Z}} \bar{y}_z \cdot \log p_{f^*}(x_z)$. Then by Gibbs' inequality, the empirical risk can be minimized only when the probabilistic prediction of each feature $p_{f^*}(x_z)$ is equal to $\bar{y}_z$.

We now proceed to show that the above sample mean of multiple labels $\bar{y}_z$ will converge to the average distribution of these labels, even though they are non-identically distributed. Since each label is a categorical random variable, this directly follows the property of multinomial distributions (Lin et al., 2022).

Lemma 4 (Convergence of the sample mean of labels)**.** *Let $\bar{p}(\cdot|z) = \frac{1}{N}\sum_{\{i|z\in\mathscr{Z}^{(i)}\}} p_Y(\cdot|x^{(i)})$, we have with probability at least $1-\delta$,*

$$\|\bar{y}_z - \bar{p}(\cdot|z)\| \leq \tilde{O}\left(\sqrt{KN^{-1}M|\mathscr{Z}|^{-1}\delta^{-1}}\right). \tag{2.4}$$

It is also feasible to achieve the above lemma by applying Lindeberg's central limit theorem, with a weak assumption that the variance of each label is finite.

Next, we show that the average label distribution $\bar{p}(\cdot|z)$ approximates the true label distribution of the corresponding feature z.

Lemma 5 (Approximation of the true label distribution of each feature)**.**

$$\|\bar{p}(\cdot|z) - p_Y(\cdot|z)\| \leq \tilde{O}\left(M|\mathscr{Z}|^{-1}\right). \tag{2.5}$$

The intuition is that since the average distributions $\bar{p}(\cdot|z)$ here are all contributed by the true label distribution of the inputs that contain feature z, it will be dominated by the label distribution of feature z, given some minor assumption on the sampling process of features when generating each input.

Finally, combining Lemmas 3, 4, 5, one can see that the probabilistic prediction of each feature given by the empirical risk minimizer will approximate the true label distribution of that feature. Subsequently, we can show that the probabilistic prediction of each input approximates the true label distribution of that input, which leads to the following main theorem.

Theorem 2 (Approximation error under a hypothetical case)**.** *Given the setup introduced in Section 2.3.1, let f^* be a minimizer of the empirical risk (Eq. (2.1)) and assume f_E^* is an invariant feature extractor. Then for any input $x \in \mathscr{D}$, with probability at least $1-\delta$,*

$$\|f^*(x) - p^*(x)\| \leq \tilde{O}\left(\sqrt{KN^{-1}M|\mathscr{Z}|^{-1}\delta^{-1}}\right) + \tilde{O}\left(M|\mathscr{Z}|^{-1}\right). \tag{2.6}$$

2.3.3 Realistic Case

We now show that in a realistic case where the feature extractor is not exactly invariant, it is still possible to approximate the true label distribution, as long as the feature extractor is robust to the variation of features across inputs and also robust to feature transformations.

Definition 4 (Lipschitz-continuous feature extractor)**.** *We call f_E a L_X-Lipschitz-continuous feature extractor, if for any two inputs i and j, $\|f_E(x_m^{(i)}) - f_E(x_{m'}^{(j)})\| \leq L_X \|x_m^{(i)} - x_{m'}^{(j)}\|$, as long as $z_m = z_{m'}$, namely the feature names of the patches m in i and patch m' in j match.*

Definition 5 (Transformation-robust feature extractor)**.** *We call f_E a L_Γ-transformation-robust feature extractor, if for any patch m in any input i, and for any transformation γ, $\|f_E(\gamma(x_m^{(i)})) - f_E(x_m^{(i)})\| \leq L_\Gamma$.*

Similar to Lemma 3, given a Lipschitz-continuous and transformation-robust feature extractor, the probabilistic predictions of features will still converge to the sample mean of the

labels where the corresponding inputs contain this feature, up to some constant error. This requires the assumption that the input representation of the same feature is similar across different inputs, which is intuitive and covered by Definition 2. All other lemmas introduced in the hypothetical case still hold trivially as they are not dependent on the network. Therefore, we can have the following result.

Theorem 3 (Approximation error under a realistic case)**.** *Given the setup introduced in Section 2.3.1, let f^* be a minimizer of the empirical risk (Eq. (2.1)) and assume f_E^* is a L_X-Lipschitz-continuous and L_Γ-transformation-robust feature extractor. Let $\nu = \max_z \nu_z$. Then for any input $x \in \mathscr{D}$, with probability at least $1-\delta$,*

$$\|f^*(x) - p^*(x)\| \leq \tilde{O}\left(\sqrt{KN^{-1}M|\mathscr{Z}|^{-1}\delta^{-1}}\right) + \tilde{O}\left(M|\mathscr{Z}|^{-1}\right) + \tilde{O}(L_X\tilde{\delta}^{-0.5}\nu) + \tilde{O}(L_\Gamma). \quad (2.7)$$

2.4 *SoTeacher*

Based on our theoretical findings, we investigate practical techniques for training the teacher model to more accurately approximate the true label distribution of the training data.

Lipschitz regularization. Theorem 3 suggests that it is necessary to enforce the feature extractor to be Lipschitz continuous. This may also be achieved by current teacher training practice, as neural networks are often implicitly Lipschitz bounded (Bartlett et al., 2017). Nevertheless, we observe that explicit Lipschitz regularization (LR) can still help in multiple experiments (See Sect. 3.4). Therefore, we propose to incorporate a global Lipschitz constraint into teacher's training. For the implementation details, we follow the existing practice of Lipschitz regularization (Yoshida & Miyato, 2017; Miyato et al., 2018) and defer them to Appendix B.3.

Consistency regularization. Theorem 3 also shows that to learn the true label distribution, it is necessary to ensure the feature extractor is robust to feature transformations. This is aligned with the standard practice which employs data augmentation as regularization. However, when data is scarce, it is better to explicitly enforce the model to be robust to transformations. This is

known as consistency regularization (CR) (Laine & Aila, 2017; Xie et al., 2020; Berthelot et al., 2019; Sohn et al., 2020) that is widely used in semi-supervised learning.

Considering the training efficiency of consistency regularization, we utilize temporal ensembling (Laine & Aila, 2017), which penalizes the difference between the current prediction and the aggregation of previous predictions for each training input. In this way, the consistency under data augmentation is implicitly regularized since the augmentation is randomly sampled in each epoch. And it is also efficient as no extra model evaluation is required for an input.

To scale the consistency loss, a loss weight is often adjusted based on a Gaussian ramp-up curve in previous works (Laine & Aila, 2017). However, the specific parameterization of such a Gaussian curve varies greatly across different implementations, where more than one additional hyperparameters have to be set up and tuned heuristically. Here to avoid tedious hyperparameter tuning we simply linearly interpolate the weight from 0 to its maximum value, namely $\lambda_{\mathrm{CR}}(t) = \frac{t}{T}\lambda_{\mathrm{CR}}^{\max}$, where T is the total number of epochs.

Summary. To recap, our teacher training method introduces two additional regularization terms. The loss function can thus be defined as $\ell = \ell_{\text{Stand.}} + \lambda_{\mathrm{LR}}\ell_{\mathrm{LR}} + \lambda_{\mathrm{CR}}\ell_{\mathrm{CR}}$, where $\ell_{\text{Stand.}}$ is the standard empirical risk defined in Eq.(2.1) and λ_{LR} is the weight for Lipschitz regularization. Our method is simple to implement and incurs only minimal computation overhead (see Section 2.5.2).

2.5 Experiments

In this section, we evaluate the effectiveness of our teacher training method in knowledge distillation. We focus on compressing a large network to a smaller one where the student is trained on the same data set as the teacher.

2.5.1 Experiment setup

We denote our method as student-oriented teacher training (DecoupledOptim), since it aims to learn the true label distribution to improve student performance, rather than to max-

imize teacher performance. We compare our method with the standard practice for teacher training in knowledge distillation (*Standard*) (*i.e.*, Eq. (2.1)). We conduct experiments on benchmark datasets including CIFAR-100 (Krizhevsky, 2009), Tiny-ImageNet (Tin, 2017), and ImageNet (Deng et al., 2009). We experiment with various backbone networks including ResNet (He et al., 2016), Wide ResNet (Zagoruyko & Komodakis, 2016) and ShuffleNet (Zhang et al., 2018b; Tan et al., 2019). We test the applicability of DecoupledOptimfrom different aspects of model compression in knowledge distillation including reduction of the width or depth, and distillation between heterogeneous neural architectures.

For knowledge distillation algorithms, we experiment with the original knowledge distillation method (KD) (Hinton et al., 2015), and a wide variety of other sophisticated knowledge distillation algorithms (see Sect. 2.5.2). We report the classification accuracies on the test set of both teacher and the student distilled from it. All results except ImageNet are presented with mean and standard deviation based on 3 independent runs. For Tiny-ImageNet, we also report the top-5 accuracy. For hyperparameters, we set $\lambda_{\mathrm{CR}}^{\mathrm{max}} = 1$ for both datasets, $\lambda_{\mathrm{LR}} = 10^{-5}$ for CIFAR-100 and $\lambda_{\mathrm{LR}} = 10^{-6}$ for Tiny-ImageNet/ImageNet. More detailed hyperparameter settings for neural network training and knowledge distillation can be found in Appendix B.4.

Table 2.1. Test accuracy of the teacher and student with knowledge distillation conducted on CIFAR-100. DecoupledOptimachieves better student accuracy than Standard for various architectures, depsite a lower teacher accuracy.

	WRN40-2/WRN40-1		WRN40-2/WRN16-2		ResNet32x4/ShuffleNetV2	
	Student	Teacher	**Student**	Teacher	**Student**	Teacher
Standard	73.73 ± 0.13	76.38 ± 0.13	74.87 ± 0.45	76.38 ± 0.13	74.86 ± 0.18	79.22 ± 0.03
DecoupledOptim	$\mathbf{74.35} \pm 0.23$	74.95 ± 0.28	$\mathbf{75.39} \pm 0.23$	74.95 ± 0.28	$\mathbf{77.24} \pm 0.09$	78.49 ± 0.09
no-CR	74.34 ± 0.11	74.30 ± 0.12	75.20 ± 0.24	74.30 ± 0.12	76.52 ± 0.52	77.73 ± 0.17
no-LR	73.81 ± 0.15	76.71 ± 0.16	75.21 ± 0.13	76.71 ± 0.16	76.23 ± 0.18	80.01 ± 0.18

2.5.2 Results

End-to-end Knowledge Distillation Performance. Tables 2.1 and 2.2 show the evaluation results on CIFAR-100 and Tiny-ImageNet/ImageNet, respectively. Our teacher training method

Table 2.2. Test accuracy of the teacher and student with knowledge distillation conducted on Tiny-ImageNet and ImageNet. The student network is ResNet18, while the teacher network is ResNet34 for Tiny-ImageNet and ResNet152 for ImageNet. Due to computation constraints, we are not able to perform ablation experiments on ImageNet.

	Tiny-ImageNet				ImageNet	
	Student (Top-1)	**Student** (Top-5)	Teacher (Top-1)	Teacher (Top-5)	**Student** (Top-1)	Teacher (Top-1)
Standard	66.19 ± 0.17	85.74 ± 0.21	64.94 ± 0.32	84.33 ± 0.40	71.30	77.87
DecoupledOptim	$\mathbf{66.83} \pm 0.20$	86.19 ± 0.22	64.88 ± 0.48	84.91 ± 0.41	71.45	77.11
no-CR	66.39 ± 0.27	86.05 ± 0.17	64.36 ± 0.43	84.10 ± 0.27	-	-
no-LR	66.48 ± 0.43	$\mathbf{86.20} \pm 0.40$	64.26 ± 1.54	84.48 ± 0.84	-	-

DecoupledOptim can improve the student's test accuracy consistently across different datasets and teacher/student architecture pairs. Note that the success of our teacher training method is not due to the high accuracy of the teacher. In Tables 2.1 and 2.2, one may already notice that our regularization method will hurt the accuracy of the teacher, despite that it can improve the accuracy of the student distilled from it.

On ImageNet particularly, the gain in student's accuracy by DecoupledOptim is minor, potentially because the large size of the dataset can already facilitate the learning of true label distribution as suggested by our theory. Nevertheless, one may observe that the teacher's accuracy is significantly lower when using DecoupledOptim, which implicitly suggests our teacher regularization method is tailored towards the student's accuracy.

Table 2.3. Estimation of the uncertainty quality of the teacher network trained by Standard and DecoupledOptim. The uncertainty quality is estimated by the ECE and NLL both before and after temperature scaling (TS).

Dataset	Teacher	Method	ECE	NLL	ECE (w/ TS)	NLL (w/ TS)
CIFAR-100	WRN40-2	Standard	0.113 ± 0.003	1.047 ± 0.007	0.028 ± 0.004	0.905 ± 0.008
		DecoupledOptim	0.057 ± 0.002	0.911 ± 0.013	0.016 ± 0.003	0.876 ± 0.012
CIFAR-100	ResNet32x4	Standard	0.083 ± 0.003	0.871 ± 0.010	0.036 ± 0.001	0.815 ± 0.008
		DecoupledOptim	0.037 ± 0.001	0.777 ± 0.006	0.021 ± 0.001	0.764 ± 0.003
Tiny-ImageNet	ResNet34	Standard	0.107 ± 0.007	1.601 ± 0.037	0.043 ± 0.002	1.574 ± 0.015
		DecoupledOptim	0.070 ± 0.007	1.496 ± 0.031	0.028 ± 0.002	1.505 ± 0.033

Ablation Study. We toggle off the Lipschitz regularization (LR) or consistency regularization (CR) in DecoupledOptim(denoted as *no-LR* and *no-CR*, respectively) to explore their individual

effects. As shown in Tables 2.1 and 2.2, LR and CR can both improve the performance individually. But on average, DecoupledOptimachieves the best performance when combining both LR and CR, as also demonstrated in our theoretical analyses. Note that in Table 2.2, using Lipschitz regularization is not particularly effective because the regularization weight might not be properly tuned (see Figure 2.2).

Quality of True Distribution Approximation. To further interpret the success of our teacher training method, we show that our regularization can indeed improve the approximation of the true label distribution thus benefiting the student generalization. Directly measuring the quality of the true distribution approximation is infeasible as the true distribution is unknown for realistic datasets. Follow previous works (Menon et al., 2021), we instead *estimate* the approximation quality by reporting the Expected Calibration Error (ECE) (Guo et al., 2017a) and NLL loss of the teacher on a holdout set with one-hot labels. Since scaling the teacher predictions in knowledge distillation can improve the uncertainty quality (Menon et al., 2021), we also report ECE and NLL after temperature scaling, where the optimal temperature is located on an additional holdout set (Guo et al., 2017a). As shown in Table 2.3, our teacher training method can consistently improve the approximation quality for different datasets and teacher architectures.

Effect of Hyperparameters. We conduct additional experiments on Tiny-ImageNet as an example to study the effect of two regularization terms introduced by our teacher training method. For Lipschitz regularization, we modulate the regularization weight λ_{LR}. For consistency regularization, we try different maximum regularization weights $\lambda_{\text{CR}}^{\text{max}}$ and different weight schedulers including linear, cosine, cyclic, and piecewise curves. Detailed descriptions of these schedulers can be found in Appendix B.4. As shown in Figure 2.2, both Lipschitz and consistency regularizations can benefit the teacher training in terms of the student generalization consistently for different hyperparameter settings. This demonstrates that our regularizations are not sensitive to hyperparameter selection. Note that the hyperparameter chosen to report the results in Tables 2.1 and 2.2 might not be optimal since we didn't perform extensive hyperparameter search

in fear of overfitting small datasets. It is thus possible to further boost the performance by careful hyperparameter tuning.

In particular, Figure 2.2(a) shows that, as Lipschitz regularization becomes stronger, the teacher accuracy constantly decreases while the student accuracy increases and converges. This demonstrates that excessively strong Lipschitz regularization hurts the performance of neural network training, but it can help student generalization in the knowledge distillation context.

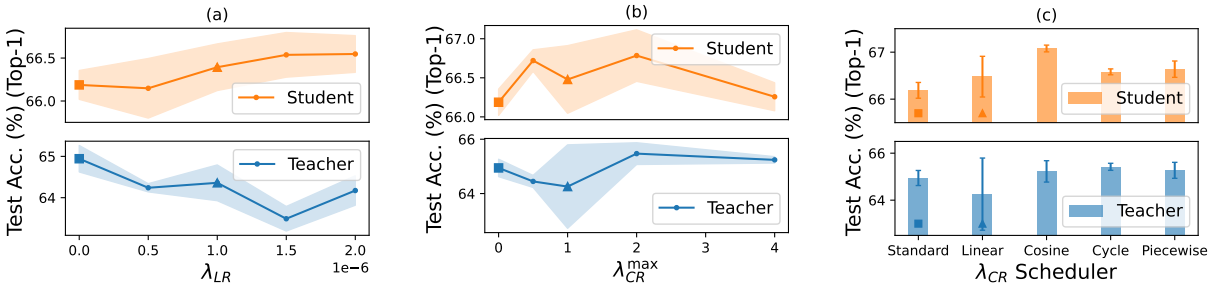


Figure 2.2. Effect of varying the hyperparameters in our teacher training method, including the weight for Lipschitz regularization λ_{LR}, the weight for consistency regularization λ_{CR}, and its scheduler. The settings of our method used to report the results (e.g. Table 2.2) are denoted as "▲". The standard teacher training practice is denoted as "■" for comparison.

Other Knowledge Distillation Algorithms. Besides the original knowledge distillation algorithm, we experiment with various feature distillation algorithms including FitNets (Romero et al., 2015), AT (Zagoruyko & Komodakis, 2017), SP (Tung & Mori, 2019), CC (Peng et al., 2019), VID (Ahn et al., 2019), RKD (Park et al., 2019), PKT (Passalis & Tefas, 2018), AB (Heo et al., 2019), FT (Kim et al., 2018), NST (Huang & Wang, 2017), CRD (Tian et al., 2020) and SSKD (Xu et al., 2020a). For the implementation of these algorithms, we refer to existing repositories for knowledge distillation (Tian et al., 2020; Shah et al., 2020; Matsubara, 2021) and author-provided codes. Although these distillation algorithms match all types of features instead of predictions between teacher and student, they will achieve the best distillation performance when combined with the prediction distillation (*i.e.* original KD). Therefore, our teacher training method should still benefit the effectiveness of these distillation algorithms. We also experiment with a curriculum distillation algorithm RCO (Jin et al., 2019) which distills from multiple

checkpoints in teacher's training trajectory. Our teacher training method should also benefit RCO as the later teacher checkpoints become more student-oriented. As shown in Table B.2, our DecoupledOptimcan boost the distillation performance of almost all these distillation algorithms, demonstrating its wide applicability.

Student fidelity. Recent works have underlined the importance of student fidelity in knowledge distillation, namely the ability of the student to match teacher predictions (Stanton et al., 2021). Student fidelity can be viewed as a measure of knowledge distillation effectiveness that is orthogonal to student generalization, as the student is often unable to match the teacher predictions although its accuracy on unseen data improves (Furlanello et al., 2018; Mobahi et al., 2020). Here we measure the student fidelity by the average agreement between the student and teacher's top-1 predicted labels on the test set. As shown in Table 2.4, our teacher training method can consistently and significantly improve the student fidelity for different datasets and teacher-student pairs, which aligns with the improvement of student generalization shown by Table 2.1 and 2.2. This demonstrates that the teacher can better transfer its "knowledge" to the student with our training method.

Table 2.4. Average agreement (%) between the student and teacher's top-1 predictions on the test set.

	CIFAR-100			Tiny-ImageNet
	WRN-40-2/WRN-40-1	WRN-40-2/WRN-16-2	ResNet32x4/ShuffleNetV2	ResNet34/ResNet18
Standard	76.16 ± 0.14	76.92 ± 0.29	76.63 ± 0.25	71.33 ± 0.07
DecoupledOptim	$\mathbf{77.92} \pm 0.27$	$\mathbf{79.41} \pm 0.11$	$\mathbf{80.36} \pm 0.13$	$\mathbf{73.36} \pm 0.25$

2.6 Related Work

Understand knowledge distillation. There exist multiple perspectives in understanding the effectiveness of knowledge distillation. Besides the statistical perspective which views the soft prediction of the teacher as an approximation of the true label distribution (Menon et al., 2021; Dao et al., 2021), another line of work understands knowledge distillation from a regularization

perspective, which views the teacher's soft predictions as instance-wise label smoothing (Yuan et al., 2020; Zhang & Sabuncu, 2020; Tang et al., 2020). More recently, (Allen-Zhu & Li, 2020) understands knowledge distillation from a feature learning perspective by focusing on the data that possesses a "multi-view" structure, that multiple features co-exist in the input and can be used to make the correct classification. Knowledge distillation is effective because the teacher can learn different features and transfer them to the student. Our theory is built on a similar assumption on the data structure, albeit we require the teacher to learn the true label distribution of the feature. Our theory thus can also be viewed as a bridge between the statistical perspective and feature learning perspective.

Alleviate "teacher overfitting". Since in knowledge distillation, the distillation data is often reused from the teacher's training data, a teacher trained toward convergence is very likely to overfit its soft predictions on the distillation data. Intuitively, it is possible to tackle this problem by early stopping the teacher training (Cho & Hariharan, 2019). However, a meticulous hyperparameter search may be required since the epoch number to find the best checkpoint is often sensitive to the specific training setting such as the learning rate schedule. It is also possible to save multiple early teacher checkpoints for the student to be distilled from sequentially (Jin et al., 2019). Additionally, one can utilize a "cross-fitting" procedure to prevent the teacher from memorizing the training data. Namely, the training data is first partitioned into several folds, where the teacher predictions on each fold are generated by the teacher trained only on out-of-fold data (Dao et al., 2021). One can also train the teacher network jointly with student's network blocks, which imposes a regularization toward the student performance (Park et al., 2021). Different from these attempts, we train the teacher to directly learn the true label distribution of its training data, leading to a simple and practical student-oriented teacher training framework with minimum computation overhead.

2.7 Summary

In this chapter, we rigorously studied the feasibility to learn the true label distribution of the training data under a standard empirical risk minimization framework. We also explore possible improvements of current teacher training that facilitate such learning. In the future, we plan to adapt our theory to other knowledge distillation scenarios such as transfer learning and mixture of experts, and explore more effective student-oriented teacher network training methods.

This chapter incorporates material from the publication "Toward Student-oriented Teacher Network Training for Knowledge Distillation" by Chengyu Dong, Liyuan Liu, and Jingbo Shang, published in The Twelfth International Conference on Learning Representations (ICLR 2024). The dissertation author was the primary investigator and lead author of this paper.

Chapter 3

Rule-Generated Data

The previous chapters explored how model-generated data can enhance training through both adversarial and supportive approaches. While these methods demonstrate significant potential, they still require high-quality labeled data to train the auxiliary models that generate the training signals. In many real-world scenarios, obtaining such labeled data can be prohibitively expensive or time-consuming, especially for specialized domains or new tasks.

An alternative approach to scaling up training data without extensive human annotation is to leverage rule-generated data. Instead of using models to generate training signals, this approach employs human-designed rules or heuristics to automatically annotate data. For instance, in text classification, domain experts can specify keywords or patterns that are indicative of certain categories. These rules can then be applied to large amounts of unlabeled text to create training data efficiently.

However, rule-generated data presents its own unique challenges. Unlike model-generated data, where the auxiliary models can potentially learn complex patterns from the training data, rule-based annotation typically relies on simple, explicit patterns. This simplicity, while making the rules easy to design and apply, can lead to strong biases in the generated training data. Models trained on such data may overly rely on these simple patterns rather than learning more robust and generalizable features.

In this chapter, we focus on one of the most widely used approaches for generating

rule-based training data for text classification: seed-based weak supervision. While this method is appealingly simple - using keyword matching to automatically label texts - its effectiveness has been historically limited by the biases introduced during the labeling process. We demonstrate that by carefully considering and addressing these biases during model training, we can significantly improve the utility of rule-generated training data.

3.1 Introduction to Debiasing Made State-of-the-art: Revisiting the Simple Seed-based Weak Supervision for Text Classification

Recently, weakly supervised text classification, because of its light requirement of human effort, has been extensively studied (Mekala & Shang, 2020; Wang et al., 2021; Zhao et al., 2022; Meng et al., 2020; Zhang et al., 2021; Meng et al., 2018; Tao et al., 2015; Park & Lee, 2022). Specifically, it requires only high-level human guidance to label the text, such as a few rules provided by human experts that match the text with the labels. These labels, which are not necessarily correct and are thus often dubbed as pseudo-labels, are then employed to train the text classifier following a standard fully supervised or semi-supervised training framework. State-of-the-art methods mostly focus on designing sophisticated human guidance to obtain high-quality labels, through contextualized weak supervision (Mekala & Shang, 2020), prompting language models (Meng et al., 2020; Zhao et al., 2022), clustering for soft matching (Wang et al., 2021), and complicated interactions between seeds (Zhang et al., 2021).

In this paper, we revisit the seed matching-based weak supervision (denoted as **Vanilla**) (Mekala & Shang, 2020; Meng et al., 2018; Tao et al., 2015), which is arguably the simplest way to generate pseudo-labels, and show that its power was greatly underestimated. Specifically, this simple method matches input text with a label if the user-provided seed words of this label are contained in the input text. For example, in sentiment analysis, a document will be labeled as "positive" if it contains the word "happy". A text classifier is then trained based on all these pseudo-labels.

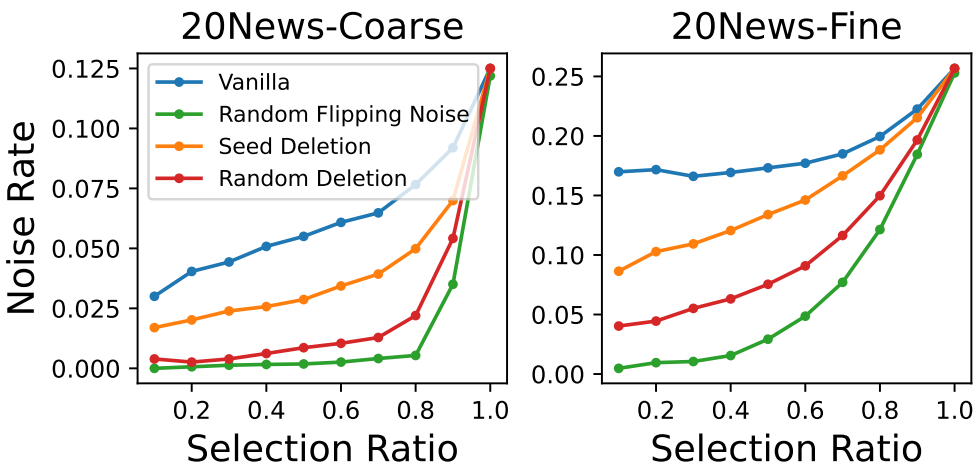


Figure 3.1. Noise rate in the subset of pseudo-labels selected based on the confidence score of a classifier trained on the pseudo-labeled data. We show the noise rates at multiple selection ratios. ("Vanilla"): the classifier is trained on the original pseudo-labeled data given by seed matching. ("Random Flipping Noise"): the classifier is trained on noisy pseudo-labeled data synthesized using the ground-truth labels, where the data examples, the overall noise rate, noise rate in each class, and the noise transition rate between any two classes are identical to the label noise induced by seed matching. ("Seed Deletion"): The classifier is trained on pseudo-labeled data where the seeds are deleted from the text. (Random Deletion): The classifier is trained on pseudo-labeled data where the words in the text are randomly deleted.

One can expect a non-trivial number of errors in the seed matching-based pseudo-labels. In an ideal case, if we can select only those correct pseudo-labels for training, the accuracy of the text classifier can be significantly boosted. For example, on the 20 Newsgroups dataset, ideally with only those correct pseudo-labels one can get an accuracy of 90.6%, compared to 80.1% obtained on all pseudo-labels (see more in Section 3.4). In practice, to select those correct labels, a common way is to use the confidence score of a classifier trained on pseudo-labels (Rizve et al., 2021). However, those high-confidence pseudo-labels may not be correct in the weakly-supervised setting, likely because the classifier may fail to learn reliable confidence on these noisy pseudo-labels (Mekala et al., 2022).

In this paper, we take a deep dive into this problem and find that, surprisingly, the high noise rate among the pseudo-labels is often not an obstacle to learning reliable confidence at all. In fact, on a set of synthesized pseudo-labels where the noise rate is exactly the same as those given by seed matching, but the noisy labels are generated by **randomly flipping** true labels into

other classes, the confidence learned by a classifier can genuinely reflect the correct labels, as shown in Figure 3.1.

Therefore, we argue that the poor confidence learned on realistic pseudo-labels is largely attributed to the strong but likely erroneous correlation between the text and pseudo-label injected by the seed-matching rule, which we refer to as *label bias*. Such a bias can be easily learned by the text classifier upon training, thus yielding spuriously high confidence on any text matching the seed word and ruining the pseudo-label selection.

To defend against such a label bias, we propose to simply delete the seed words present in the text upon training a classifier on the pseudo-labeled data, which effectively prevents the classifier from learning the biased correlation between seeds and the corresponding pseudo-labels. As shown in Figure 3.1, such a simple **seed deletion** method can significantly improve the confidence score of the trained classifier and thus help select pseudo-labels with fewer label errors at every selection ratio. Empirical results verify that these less noisy pseudo-labels can indeed improve the classification accuracy significantly, making seed matching-based weak supervision on par with or sometimes even better than the state-of-the-art.

We further investigate the scenario where the seed words are *not* made known. We propose to delete every word token in the input text randomly and independently. This simple **random deletion** method can improve confidence learning even more as shown in Figure 3.1. Our theoretical analysis also shows that this random deletion method can mitigate the label bias with a high probability and therefore recover the seed deletion in effect. It is worth noting that both of these methods introduce no additional hyperparameters.

In summary, our contributions are as follows.

- We revisit the seed matching-based weak supervision and find that its effectiveness is mainly limited by the label bias injected by the seed-matching rule.
- We show that simply deleting seed words from the pseudo-labeled texts can significantly alleviate the label bias and improve the confidence estimation for pseudo-label selection, as well as end-to-end classification accuracy achieved by seed matching, on par with or even

better than the state-of-the-art.

- We further propose the random deletion method to handle the case when the seed words are unknown and demonstrate its effectiveness both empirically and theoretically.

3.2 Preliminaries and Related Work

Seed matching as simple weak supervision. Seed matching (Meng et al., 2018) is probably one of the simplest weak supervision. Specifically, for each label class, a user provides a set of seed words that are indicative of it. A given document is annotated as the label class whose seed words appear in the document, or annotated as the label class whose seed words appear most frequently if multiple such label classes exist. Sophisticated weak supervisions have also been proposed to generate pseudo-labels with better quality, such as meta data (Mekala et al., 2020), context (Mekala & Shang, 2020), sentence representation (Wang et al., 2021), predictions of masked language models (Meng et al., 2020) and keyword-graph predictions (Zhang et al., 2021).

Confidence learning matters for pseudo-label selection. Since label errors prevail in the pseudo-labels generated by weak supervision, it is often necessary to select the pseudo-labels before incorporating them into training (Wang et al., 2021; Mekala et al., 2022). A common method to select those pseudo-labels is to use the model confidence, namely the probability score associated with a deep classifier's prediction, to determine whether a given pseudo-label is correct or not (Guo et al., 2017b). However, such confidence often cannot genuinely reflect the correctness of the pseudo-label generated by weak supervision, in that pseudo-labels with high confidence are not necessarily correct (Mekala et al., 2022)

Backdoor attack and defense. A problem related to seed matching is the backdoor attack for text classification based on trigger words (Dai et al., 2019; Kurita et al., 2020; Chen et al., 2021). Such attacks corrupt a dataset by inserting a particular word (*i.e.*, trigger) into several documents and change their corresponding labels as ones specified by the attacker. A model fine-tuned

on such a dataset will predict the label specified by the attacker whenever a document contains the trigger word. This is largely because the model would overfit the malicious correlation between the trigger word and the specified label, which is similar to the problem when learning pseudo-labels generated by seed matching. Therefore, trigger-based backdoor attacks may be defended by the methods proposed in this work as well, especially random deletion since the attacker will not reveal the trigger word.

3.3 Method

3.3.1 Seed deletion

We describe the details of seed deletion. Specifically, we denote an input document composed of a set of words as $x = \{t_1, t_2, \cdots, t_n\}$ and its pseudo-label given by seed matching as $\tilde{y}$. We denote the set of seed words from class y as $\mathscr{S}_y$. Now for each document in the pseudo-labeled dataset, we generate a corrupted document $\hat{x}$ by deleting any seed word in x that is associated with its pseudo-label, namely $\hat{x} = \{t | t \in x, t \notin \mathscr{S}_{\tilde{y}}\}$. We then train a classifier $\hat{\theta}$ on the corrupted dataset $\hat{D} = \{(\hat{x}, \tilde{y})\}$ and use its confidence score at the pseudo-label $P_{\hat{\theta}}(\tilde{y}|x)$ as an uncertainty measure to select the correct pseudo-labels.

Note that when generating the uncertainty measure on a (document, pseudo-label) pair, one can either evaluate the classifier on the original document or the corrupted document. Empirically we found that evaluating the classifier on the original document would produce a minor gain.

3.3.2 Random deletion

In real-world applications, the seed words provided by the user may not always be accessible due to privacy concerns or simply because they are lost when processing and integrating the data. We show that it is still feasible to perform seed deletion in a probabilistic manner without knowing seed words, while remaining effective for confidence-based pseudo-label selection.

To achieve this, in fact, we only have to delete the words randomly and independently in a given document. Specifically, give a deletion ratio p, for every document $x = \{t_1, t_2, \cdots, t_n\}$, we randomly sampled a few positions $\mathscr{M} = \{i_1, i_2, \cdots, i_{\lceil pn \rceil}\}$, where $\lceil \cdot \rceil$ denotes the ceiling of a number. We then generate a corrupted document by deleting words at those positions, namely $\hat{x} = \{t_i | i \in \{1, 2, \cdots, n\}, i \notin \mathscr{M}\}$. Now on the corrupted dataset $\mathscr{D} = \{(\hat{x}, \tilde{y})\}$, we can train a classifier $\hat{\theta}$ and utilize its confidence score $P_{\hat{\theta}}(\tilde{y}|x)$ for pseudo-label selection, similar to seed deletion.

Random deletion as a probabilistic seed deletion. Despite its simplicity, we show with high probability, random deletion can mitigate the label bias induced by seed matching. The intuition here is that since the document only contains one or a few seed words, by random deletion it is very likely we can delete all seed words while retaining at least some other words that can still help learning.

Specifically, we consider a particular type of corrupted document $\hat{x}$ that contains no seed word, *i.e.*, $\hat{x} \cap \mathscr{S}_{\tilde{y}} = \emptyset$, but contains at least one word that is indicative of the true class, *i.e.*, $\hat{x} \cap \mathscr{C}_y \neq \emptyset$, where $\mathscr{C}_y$ denotes the set of words that are indicative of the class y. Since such a document no longer contains the seed word, its pseudo-label is not spuriously correlated with the text. At the same time, it contains class-indicative words that can help the classifier learn meaningful features.

We then investigate the probability that a document becomes such a particular type after random deletion. We term such probability as the seed-deletion rate r_{SD}, which is defined as

$$r_{\text{SD}} := P(\mathbf{1}(\hat{x} \cap \mathscr{S}_{\tilde{y}} = \emptyset, \hat{x} \cap \mathscr{C}_y \neq \emptyset)), \tag{3.1}$$

where $\mathbf{1}(\cdot)$ is the indicator function. In an ideal case where $r_{\text{SD}} = 1$, we can completely recover the effect of seed deletion on eliminating label bias.

Now since each word in the document is independently deleted, we have

$$r_{\mathrm{SD}} = p^{n_s} \cdot (1 - p^{n_c}), \tag{3.2}$$

where $n_s \coloneqq |\mathscr{S}_{\tilde{y}}|$ denotes the number of seed words in the document and $n_c \coloneqq |\mathscr{C}_{\tilde{y}}|$ denotes the number of words in the document that are indicative of the class. One may find that when $n_c \gg n_s$, r_{SD} can be quite close to 1 as long as p is large.

Estimate the best deletion ratio. We estimate the best deletion ratio for random deletion based on some reasonable assumptions. First, it is easy to see that based on Eq. (3.2), the optimal deletion ratio is

$$p^* = \left(\frac{n_s}{n_s + n_c}\right)^{\frac{1}{n_c}}, \tag{3.3}$$

which depends on both the number of seed words n_s and the number of class-indicative words n_c in the document. For n_s, we can simply set it as 1 since the pseudo-label of a document is usually determined by one or two seed words. For n_c, we assume that all words in a document are indicative of the true class, except stop words and punctuation. These estimations are acceptable as p^* is almost always close to 1 and is quite robust to the change of n_s and n_c as long as n_c is large (See Figure 3.2). This condition is likely to be true for realistic datasets (See Table 3.1). Note that for simplicity, we set one single deletion ratio for a specific dataset. Thus we set n_c as the median number of class-indicative words over all documents in a dataset.

3.4 Experiments

3.4.1 Experiment setup

We evaluate the performance of seed word matching equipped with seed deletion or random deletion on text classification.

Datasets. We report the text classification performance on the following datasets, including New York Times (*NYT*), 20 Newsgroups (*20News*), *AGNews* (Zhang et al., 2015), Rotten

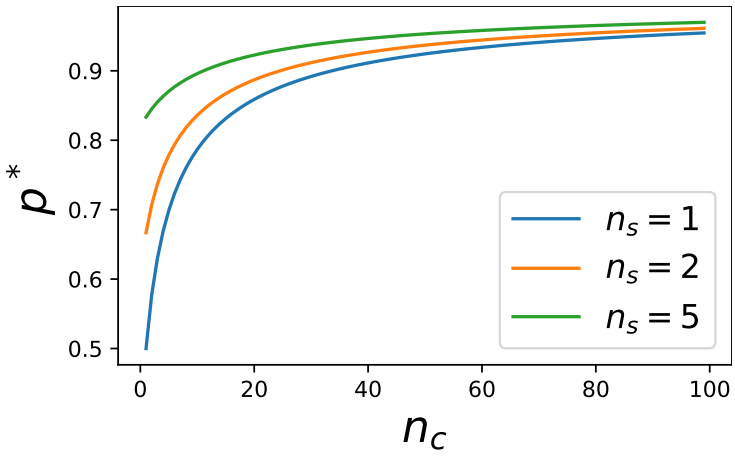


Figure 3.2. The optimal deletion ratio p^* for random deletion with respect to the number of seed words n_s and the number of class-indicative words n_c based on Eq. (3.3).

Table 3.1. Statistics of the dataset and the corresponding pseudo-labels given by seed matching. For sequence length, we report the median across all pseudo-labeled documents to reduce the impact of outliers, along with the median absolute deviation in the bracket.

Dataset	# Docs	# Labels	# Pseudo-labeled Docs	Pseudo-label Noise Rate (%)	Sequence Length (Pseudo-labeled Docs)	n_c (Estimated) (Pseudo-labeled Docs)
AGNews	120,000	4	32,359	16.26	39(6)	27(4)
20News-Coarse	17,871	5	7,671	12.50	286(136)	140(65)
NYT-Coarse	13,081	5	9,460	11.47	922(222)	462(105)
20News-Fine	17,871	17	10,455	25.67	257(120)	129(58)
NYT-Fine	13,081	26	8,229	31.80	940(214)	467(100)
Rotten-Tomatoes	10,662	2	990	28.38	26(8)	13(4)

Table 3.2. Classification performance achieved by vanilla seed matching and seed matching equipped with various pseudo-label selection methods. † indicates methods that are not fair comparison and are listed only for reference. We conduct each experiment for 5 times, report the average, and denote the standard deviation in the bracket. We report both Macro-F1 and Micro-F1 for classification accuracy.

	AGNews		**20News-Coarse**		**NYT-Coarse**		**20News-Fine**		**NYT-Fine**		**Rotten-Tomatoes**	
Method	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
Oracle†	86.3(0.3)	86.2(0.3)	90.6(0.4)	90.6(0.4)	97.0(0.2)	93.9(0.4)	81.1(0.2)	81.1(0.3)	96.5(0.1)	92.0(0.4)	79.6(1.0)	79.6(1.0)
Vanilla	83.9(0.6)	83.9(0.6)	80.5(0.6)	80.1(0.6)	87.8(0.2)	78.4(0.4)	68.2(0.6)	69.0(0.7)	73.0(0.7)	68.7(0.8)	71.4(1.2)	71.3(1.2)
Standard Confidence	82.2(2.1)	82.0(2.1)	78.9(1.8)	80.0(1.5)	88.3(4.1)	79.1(2.8)	64.4(2.2)	66.6(1.8)	45.8(0.5)	58.6(0.3)	72.2(2.4)	72.0(2.4)
O2U-Net	79.8(0.5)	79.8(0.5)	80.9(0.3)	78.5(0.2)	92.9(0.4)	85.9(0.7)	71.1(0.4)	71.2(0.8)	14.7(10.2)	8.70(7.3)	**74.1(1.4)**	**74.0(1.4)**
LOPS	79.5(0.9)	79.5(0.6)	81.7(1.0)	80.7(0.4)	**94.6(0.4)**	**88.4(0.5)**	73.8(0.6)	72.7(1.0)	84.3(0.5)	**81.6(0.3)**	70.4(0.4)	70.4(0.4)
Seed-Deletion	84.3(0.7)	84.2(0.7)	**86.4(0.9)**	**86.1(0.8)**	92.4(1.3)	85.0(2.0)	73.7(0.7)	75.0(0.5)	81.7(1.5)	79.4(1.1)	70.4(1.3)	70.3(1.3)
Random-Deletion	**86.2(0.5)**	**86.1(0.5)**	84.4(0.9)	84.8(0.8)	91.7(1.3)	83.3(1.8)	**76.3(0.8)**	**76.8(0.7)**	**84.6(1.4)**	79.6(1.1)	73.6(4.3)	73.4(4.5)
Paraphrase†	85.4(0.3)	85.4(0.3)	86.9(1.2)	86.7(1.2)	94.0(0.8)	88.5(0.7)	75.6(0.6)	76.8(0.5)	76.1(4.0)	74.8(2.4)	75.4(0.1)	75.3(0.1)
MLM-Replace†	85.8(0.1)	85.8(0.1)	87.4(0.1)	87.5(0.2)	94.5(0.1)	88.9(0.2)	73.6(1.0)	74.8(0.8)	84.1(0.6)	80.0(0.4)	76.7(1.5)	76.7(1.5)

tomatoes (Pang & Lee, 2005), as well as NYT and 20News datasets with their fine-grained labels respectively. We select these datasets as they cover diverse data properties in text classification, such as topic classification and sentiment analysis, long and short input documents, coarse-grained and fine-grained labels, and balanced and imbalanced label distributions.

For seed word matching, we consider the seed words used in (Mekala & Shang, 2020; Wang et al., 2021). Table 3.1 shows the statistics of these datasets and the corresponding pseudo-labels given by seed matching.

Training setting. We adhere to the following experiment settings for all methods unless otherwise specified. We utilize *BERT* (`bert-base-uncased`) (Devlin et al., 2019b) as the text classifier since we found that BERT can produce reliable confidence estimates with our methods in our experiments.

For pseudo-label selection, we select 50% of the high-quality pseudo-labels for all methods and datasets. For random deletion specifically, we set the deletion ratio following the estimation in Section 3.3.2. The estimated best deletion ratio of each dataset can be found in Figure 3.3.

Finally, we employ the standard self-training protocol to utilize both the documents labeled by weak supervision and additional documents that are not labeled. Note that if we selected a subset of pseudo-labels before self-training, those pseudo-labeled documents that were not selected will be merged into unlabeled documents. For the self-training process specifically, we train a text classifier on the labeled documents and generate predictions on unlabeled documents. The top τ fraction of predictions with the highest confidence are then treated as new pseudo-labels and will be merged with the existing pseudo-labels for another training. In our experiments, we conduct this self-training process for 5 iterations.

Note that one can use BERT to select high-quality pseudo-labels alone following our methods while employing advanced text classifiers for subsequent self-training, which may further improve the performance.

Comparative methods. We compare our proposed method with the following baselines.

- *Vanilla*: Self-training on all the pseudo-labeled provided by seed matching, without pseudo-label selection.
- *Standard confidence*: Train a classifier on all the pseudo-labeled documents and use its confidence score to select a subset of high-quality pseudo-labels.
- *O2U-Net* (Huang et al., 2019): Train a classifier on all the pseudo-labeled documents and use the normalized loss of each document throughout the training as a metric to select pseudo-labels.
- *LOPS* (Mekala et al., 2022): Train a classifier on all the pseudo-labeled documents and use the learning order cached during training to select pseudo-labels. We follow the setting recommended in their paper and set $\tau = 50\%$.
- *Oracle**: Based on the true labels, we select only those correct pseudo-labels for self-training. Note that this is not a realistic method and is used only for comparison.

Whenever it is necessary to train a classifier to obtain the confidence, we train for 4 epochs, in line with the setting in LOPS.

3.4.2 Main results

Seed-based weak supervision. We present the classification performance achieved by different methods in Table 3.2. One may find that seed deletion and random deletion can significantly boost the performance of seed matching-based weakly-supervised classification. On some datasets (e.g., AGNews), random deletion can approach the oracle selection with almost no gap. This demonstrates the performance of the simple seed matching-based weak supervision is greatly underestimated.

Seed deletion and random deletion are also on par with or significantly better than other pseudo-label selection methods including those using sophisticated confidence measures such as the normalized loss in O2U-Net and the learning order in LOPS. In fact, seed deletion and random deletion are still using the standard confidence score of a classifier to select pseudo-labels, albeit they first corrupt the pseudo-labeled documents for training the classifier. Nevertheless,

the performance improvement compared to the standard confidence score is huge. For example, the improvement on NYT-Fine is as large as $\sim 20\%$ in terms of Macro-F1 and $\sim 40\%$ in terms of Micro-F1. This demonstrates that confidence-based pseudo-label selection is greatly underestimated.

Table 3.3. Classification performance of text classification using a variety of weak supervisions. Results on sophisticated weak supervisions are cited from the corresponding paper. † indicates that result is reported as accuracy instead of F1-score in the original paper, while we still include it here since the dataset is class-balanced. We neglect Rotten-Tomatoes here since it is not reported in most of the listed papers.

	AGNews		20News-Coarse		NYT-Coarse		20News-Fine		NYT-Fine	
Method	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
ConWea	73.4	73.4	74.3	74.6	93.1	87.2	68.7	68.7	87.4	77.4
X-Class	82.4	82.3	58.2	61.1	**96.3**	**93.3**	70.4	70.4	86.6	74.7
LOTClass	84.9	84.7	47.0	35.0	70.1	30.3	12.3	10.6	5.3	4.1
ClassKG	**88.8**†	-	80	75	96	83	**78**	**77**	**92**	**80**
LIME	87.2	**87.2**	79.7	79.6	-	-	-	-	-	-
Seed Deletion	84.3	84.2	**86.4**	**86.1**	92.4	85.0	73.7	75.0	81.7	79.4
Random Deletion	86.2	86.1	84.4	84.8	91.7	83.3	76.3	76.8	84.6	**79.6**

Compare with sophisticated weak supervision. In Table 3.3, we compare the performance of seed word matching equipped with seed word deletion or random deletion with those methods using sophisticated weak supervision sources, listed as follows.

- *ConWea* (Mekala & Shang, 2020) uses pre-trained language models to contextualize the weak supervision in an iterative manner.
- *X-Class* (Wang et al., 2021) learns class-oriented document representations based on the label surface names. These document representations are aligned to the classes to obtain pseudo labels.
- *LOTClass* (Meng et al., 2020) obtains synonyms for the class names using pretrained language models and constructs a category vocabulary for each class, which is then used to pseudo-label the documents via string matching.
- *ClassKG* (Zhang et al., 2021) constructs a keyword graph to discover the correlation between keywords. Pseudo-labeling a document would be translated into annotating the subgraph that

represents the document.

- *LIME* (Park & Lee, 2022) combines seed matching with an entailment model to better pseudo-label documents and refine the final model via self-training.

For each method, we report the results obtained directly from the corresponding paper. If the results on some datasets are not reported in the original paper, we cite the results in follow-up papers if there are any. We found that with seed deletion or random deletion, seed word matching can be almost as good as or even better than those sophisticated weak supervisions. We thus believe seed word matching can be a competitive baseline and should be considered when developing more complicated weak supervisions.

Alternative seed-agnostic debiasing methods. We explore alternative methods to delete the seed words and mitigate the label bias in seed matching-based weak supervision, without knowing the seed words. We consider the following alternatives.

- *MLM-replace*: We randomly mask a subset of words in the document and use BERT to predict the masked words. The document is then corrupted by replacing the masked words with the predictions. This follows the idea of random deletion to delete the seed words probabilistically. Such a method is widely used in other applications (Gao et al., 2021; Clark et al., 2020).
- *Paraphrase*: We generate the paraphrase of a document using the T5 model (Raffel et al., 2019) fine-tuned on a paraphrase dataset APWS (Zhang et al., 2019). We use the publicly available implementation at (Duerr, 2021). This is a straightforward method to delete the seed words.

Since these alternative methods only serve as a reference for our main methods, we search their best hyperparameter, namely the mask ratio for MLM-replace and the token-generation temperature for paraphrase respectively. As shown in Table 3.2, these alternative methods can work as well as or better than random deletion. However, in practice, we would prefer using random deletion since it requires no extra model or knowledge source.

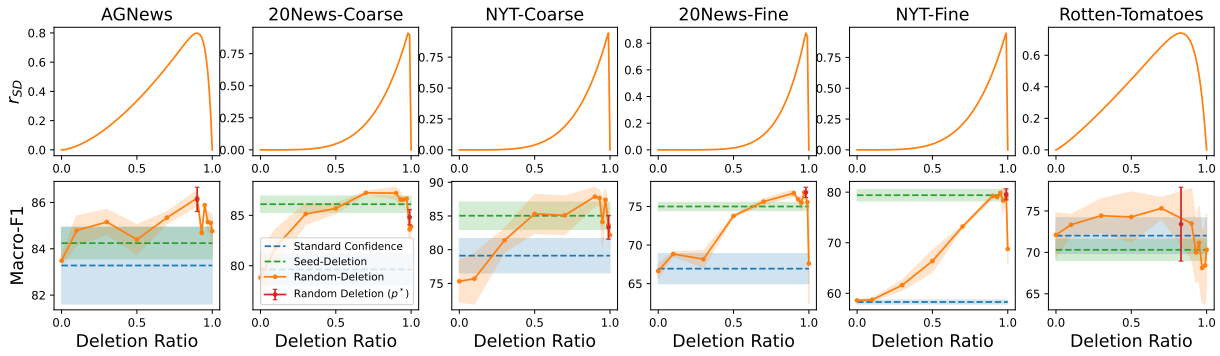


Figure 3.3. (*Top*) The seed-deletion rate r_{SD} given different deletion ratios (Eq. (3.2)), where n_s and n_c are estimated for each dataset as mentioned in Section 3.3.2. (*Bottom*) The classification performance of random deletion given different deletion ratios. "(p^*)" denotes the one using the best deletion ratio estimated for each dataset. We also denote the performance of seed deletion and standard confidence for comparison.

3.4.3 Study on random deletion

Deletion ratio in random deletion. We verify whether our estimation of the best deletion ratio is reasonable. In Figure 3.3, we modulate the deletion ratio and check the classification performance of random deletion for different datasets. One can find that as the deletion ratio increases, the performance first increases and then decreases, which is aligned with the trend of the seed-deletion rate r_{SD} analyzed in Section 3.3.2. The performance peaks when the deletion ratio is large ($\gtrsim 0.9$), which matches our estimation of the best deletion ratio. Furthermore, one may find that the best deletion ratio is relatively smaller for datasets with a shorter sequence length (e.g., AGNews and Rotten-Tomatoes), compared to that for datasets with a long sequence length (e.g., 20News and NYT), which is also predicted by our estimation.

How does random deletion work?. One may notice that in Table 3.2, random deletion can outperform seed deletion on a few datasets. This indicates that random deletion has an additional regularization effect on top of deleting seeds, potentially due to more intense data augmentation.

However, we note that random deletion works not entirely because of this additional regularization effect. To show this, we conduct ablation experiments with two additional methods. The first is random deletion but with the seed words always retained in the document. The second

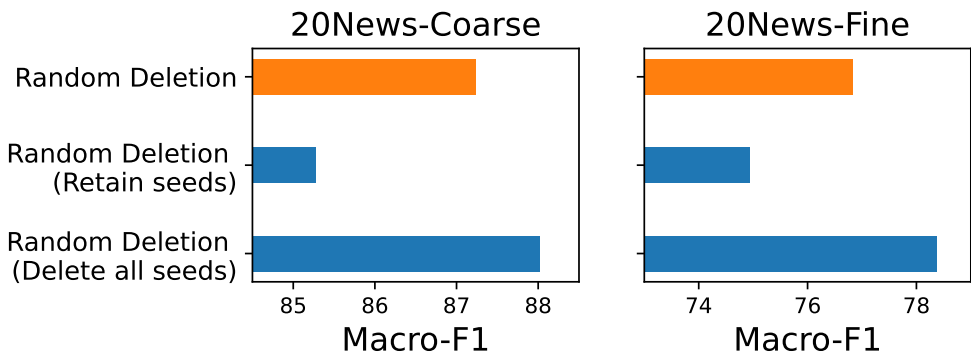


Figure 3.4. Classification performance achieved by variations of random deletion with seed words always retained in the document ("Retain seeds") and with seed words always deleted completely ("Delete all seeds").

is random deletion but with seed words first deleted completely. For a fair comparison, we search the best deletion ratio for these different methods including the standard random deletion. We experiment on two representative datasets including 20News-Coarse and 20News-Fine due to computational constraints.

Figure 3.4 shows that when seed words are always retained, random deletion achieves significantly worse performance than the standard random deletion, although the former is merely deleting one or two words fewer. On the other hand, when seed words are already deleted, further random deletion only slightly improves the performance compared to the standard random deletion. These pieces of evidence demonstrate that the benefit of random deletion may be partly attributed to a regularization effect, but the deletion of seeds and thus the mitigation of label bias is still one important factor.

Compare with additional regularization methods. Since random deletion may introduce an additional regularization effect, we compare it with other regularization methods that can potentially reduce the label bias. We will mainly compare different types of dropout (see below) that are widely used in classification tasks, since it is mostly similar to random deletion. We defer other commonly seen regularization methods to the appendix. We consider applying dropout to different positions in the transformer model. For a fair comparison, we search the best dropout ratio for these dropout methods, as well as the best deletion ratio for our random deletion. We

Table 3.4. Original document versus corrupted document after random deletion, along with the corresponding pseudo-label and true label. Here the example documents are randomly picked from the pseudo-labeled data in the 20News-Coarse dataset.

		Original Document	**Corrupted Document** (Random Deletion)
Pseudo-label	Computer	re increasing the number of serial ports distribution world nntp posting host mac4. jpl. nasa. gov in article (steven langlois) wrote does anyone know if there are any devices available for the **mac** which will increase the number of serial ports available for use simultaneously? i would like to connect up to 8 serial devices to my **mac** for an application i am working on. ...	distribution posting4 the available application working must independently. such any to the serial? ink bug only system the. them then of the. to using
Seed Word	mac		
True-label	Computer		
Pseudo-label	Computer	re dumb options list in article (charles parr) writes the idea here is to list pointless options. you know, stuff you get on a car that has no earthly use? 1) power **windows** i like my power **windows**. i think they're worth it. however, cruise control is a pretty dumb option. what's the point? if you're on a long trip, you floor the gas and keep your eyes on the rear view mirror for cops, right? power seats are pretty dumb too, unless you're unlucky enough to have to share your car. ...	the options you that).? keep are enough have like " do paper. breath. '
Seed Word	windows		
True-label	Sports		
Pseudo-label	Science	re pro abortion feminist leader endorses trashing of free speech rights in article (gordon fitch) writes (doug holtsinger) writes 51 arrested for defying judge's order at abortion protest rally the miami herald, april 11, 1993 **circuit** judge robert mcgregor's order prohibits anti abortion pickets within 36 feet of the property line of aware woman center for choice. even across the street, they may not display pictures of dead fetuses or sing or chant loud enough to be heard by patients inside the clinic. ...	leader trashying judge 11 judge abortion the display pictures loud as similar an appeal group from have a rock and then see the he homes did speech of the hear there from to expression on particular information, to others if be considered the'ists arson to else the
Seed Word	circuit		
True-label	Politics		

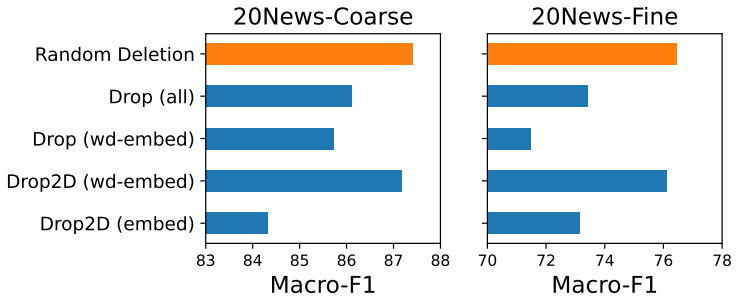


Figure 3.5. Classification performance using dropout as a regularization method for pseudo-label selection. We try two types of dropout including Dropout ("Drop") and Dropout-2D ("Drop2D"). We try dropout at various positions in the transformer architecture, including all dropouts layers in the original transformer ("all"), the embedding layer only ("embed"), and the word embedding layer only ("wd embed").

experiment on two representative datasets due to computational constraints.

- *Dropout* (Hinton et al., 2012) is a classic regularization method to prevent overfitting. We simply utilize the dropout layers built in the original transformer architecture, including those after the embedding layers, self-attention layers, and feed-forward layers, following previous work (Gao et al., 2021).
- *Dropout-2D* (Tompson et al., 2014) is different from vanilla Dropout in that it drops the entire channel as a whole. We only apply this to the embedding layer in the transformer to drop the entire embedding of a word or a position.

As shown in Figure 3.5, random deletion consistently outperforms other regularization methods. The only regularization method that can compete with random deletion is Dropout-2D applied on the word embedding layer specifically. However, one may note that this dropout variation is in fact almost equivalent to random deletion since the entire embedding of a word will be randomly dropped. These again demonstrate that random deletion works not simply because of a regularization effect.

Case study. We manually inspect the pseudo-labeled documents after random deletion to see if the label bias can be mitigated. In Table 3.4, we randomly pick some example documents after random deletion and find that the seed words are indeed deleted and some class-indicative words are still present to allow effective classification.

3.5 Conclusion and Future Work

In this chapter, we revisit the simple seed matching-based weakly supervised text classification method and show that if its pseudo-labels are properly debiased, it can achieve state-of-the-art accuracy on many popular datasets, outperforming more sophisticated types of weak supervision. Specifically, our controlled experiments show that confidence-base selection of seed matching-based pseudo-labels is ineffective largely because of the label bias injected by the simple, yet erroneous seed-match rule. We propose two effective debiasing methods, seed

deletion, and random deletion, which can mitigate the label bias and significantly improve seed matching-based weakly supervised text classification.

In future work, we plan to extend this debiasing methodology to broader problems and methods. For example, for weakly supervised text classification, we wish to explore a generalization of the debiasing strategy to more sophisticated types of weak supervision. It will also be interesting to develop a backdoor defense framework around the proposed methods, especially random deletion.

This chapter incorporates material from the publication "Debiasing Made State-of-the-art: Revisiting the Simple Seed-based Weak Supervision for Text Classification" by Chengyu Dong, Zihan Wang, and Jingbo Shang, published in The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023). The dissertation author was the primary investigator and lead author of this paper.

Chapter 4

Conclusion and Future works

4.1 Summary of Contributions

This thesis has investigated fundamental approaches to improve language model training in data-constrained scenarios. Our work spans two complementary strategies: leveraging model-generated data through adversarial and supportive models, and utilizing rule-generated data through weakly-supervised learning. The key contributions can be summarized as follows:

First, we identified and addressed a critical optimization control problem in ELECTRA-style pretraining. Our analysis revealed that the original design's coupling of generator and discriminator optimization hindered effective training. By introducing decoupled optimization, we significantly improved both training stability and model performance, demonstrating the importance of careful optimization control in adversarial learning scenarios.

Second, we provided theoretical insights into knowledge distillation, showing that it can be effective even without additional training data. Our student-oriented teacher training framework represents a paradigm shift in how we approach knowledge distillation, focusing on optimizing the teacher specifically for student performance rather than for its own accuracy. This work challenges the conventional wisdom about the relationship between teacher and student performance.

Third, we developed an effective debiasing approach for weakly-supervised learning, addressing a fundamental limitation in learning from rule-generated data. Our method significantly

improves model generalization by preventing over-reliance on simple labeling rules, making weakly-supervised learning more practical for real-world applications.

4.2 Broader Implications

Our findings have several important implications for the future of language model training:

Rethinking Data Efficiency. The success of our approaches suggests that we may be able to push the boundaries of model performance without necessarily requiring ever-larger datasets. This is particularly significant given the projected scarcity of high-quality human-generated data. Our work demonstrates that by carefully designing training mechanisms, we can make better use of existing data resources.

The Role of Auxiliary Models. Our work with both ELECTRA and knowledge distillation highlights the potential of auxiliary models in improving training efficiency. Rather than viewing these models solely as components to be optimized for their own performance, we show they can be specifically designed and trained to enhance the learning process of other models.

Scalable Training Data Generation. The improvements we achieved in weakly-supervised learning suggest a promising direction for scalable training data generation. As we approach the limits of available human-generated data, the ability to effectively leverage rule-generated data becomes increasingly important.

4.3 Future Directions

Our work opens several promising directions for future research:

Unified Training Frameworks. An exciting direction would be to develop unified frameworks that combine multiple types of generated data. For example, integrating rule-generated data with model-generated data might allow us to leverage the strengths of both approaches while mitigating their individual weaknesses.

Adaptive Generation Strategies. Future work could explore adaptive strategies for generating training data, where the generation process (whether model-based or rule-based) dynamically adjusts based on the learning progress of the target model. This could lead to more efficient and effective training procedures.

Theoretical Understanding. While we have provided theoretical analyses for specific approaches, there is still much to understand about the fundamental principles governing learning from generated data. Developing a more comprehensive theoretical framework could guide the development of even more effective training methods.

Application to Other Domains. While our work has focused on language models, the principles we've developed could potentially be applied to other domains facing similar data scarcity challenges, such as computer vision or speech recognition.

4.4 Concluding Remarks

The challenge of training effective language models with limited data resources will likely remain central to machine learning research in the coming years. This thesis has demonstrated that by carefully considering how we generate and utilize training data, we can make significant progress even within these constraints. Our work provides both theoretical insights and practical methods that we hope will contribute to the continued advancement of language models and machine learning more broadly.

As we look to the future, the approaches developed in this thesis suggest that the path forward may not lie solely in accumulating more data, but in developing smarter ways to learn from the data we have. By continuing to improve our understanding of how models learn from generated data, we can work toward more efficient and effective training methods that make the most of our available resources.

Appendix A

Model-Generated Data from "Adversarial" Models

A.1 Proof

Bound the ideal discriminator objective. Given Assumption 1, it is easy to see that the ideal discriminator objective can be bounded as $L^*(\theta_D) \leq \tilde{L}^*(\theta_D)$, where

$$\tilde{L}^*(\theta_D) := \mathbb{E}_{c,w}\mathbb{E}_{\hat{w}\sim\delta_{w|c}(\hat{w})}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w}), \tag{A.1}$$

where $\delta_{w|c}(\hat{w})$ is a Dirac measure defined by the original token, namely the discriminator loss is maximized if the original token is always sampled.

We can now proceed to prove the two Lemmas in Section 1.5.

Lemma 1.

Proof. We mainly utilize the Cauchy-Schwartz inequality, the equivalence between p-norms and Pinsker's inequality in this proof.

First, we note that,

$$L^*(\theta_D) \leq \tilde{L}^*(\theta_D) = \bar{L}_G(\theta_D) + (\tilde{L}^*(\theta_D) - \bar{L}_G(\theta_D)),$$

where the second term, the difference between two objectives, can be further bounded as

$$
\begin{aligned}
&\tilde{L}^*(\theta_D) - \bar{L}_G(\theta_D) \\
=&\mathbb{E}_{c,w}\mathbb{E}_{\hat{w}\sim\delta_{w|c}}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w}) - \mathbb{E}_{c,w}\mathbb{E}_{w\sim p_G}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w}) \\
=&\mathbb{E}_{c,w}\sum_{\hat{w}}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w})(\delta_{w|c}(\hat{w}) - p_G(\hat{w}|c)) \\
\leq&\mathbb{E}_{c,w}M_D(c,w)\left\|\delta_{w|c}(\hat{w}) - p_G(\hat{w}|c)\right\|_2 \\
\leq&\mathbb{E}_{c,w}M_D(c,w)\left\|\delta_{w|c}(\hat{w}) - p_G(\hat{w}|c)\right\|_1 \\
\leq&2^{-1/2}\mathbb{E}_{c,w}M_D(c,w)\left[D_{\mathrm{KL}}(\delta_{w|c}(\hat{w})\|p_G(\hat{w}|c))\right]^{1/2} \\
=&2^{-1/2}\mathbb{E}_{c,w}M_D(c,w)\left[H(\delta_{w|c}(\hat{w}),p_G(\hat{w}|c))\right]^{1/2} \\
\leq&2^{-1/2}m_D\left[\mathbb{E}_{c,w}H(\delta_{w|c}(\hat{w}),p_G(\hat{w}|c))\right]^{1/2} \\
=&2^{-1/2}m_D\left[\mathbb{E}_{c,w}H(\delta_{\hat{w}}(w|c),p_G(\hat{w}|c))\right]^{1/2} \\
=&2^{-1/2}m_D\left[\mathbb{E}_{c}H(q(\hat{w}|c),p_G(\hat{w}|c))\right]^{1/2} \\
=&2^{-1/2}m_D L(\theta_G)^{1/2}
\end{aligned}
\tag{A.2}
$$

where $M_D(c,w) = \|\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w})\|_2$ and $m_D = \max_{c,w} M_D(c,w)$. □

Lemma 2.

Proof. We mainly utilize a corollary of Jensen's inequality on concave functions, as well as a

simple trick that expands $\mathbb{E}[\log x]$ around $\log \mathbb{E}[x]$.

$$
\begin{aligned}
&\bar{L}_G(\theta_D) \\
&= \mathbb{E}_{c,w}\mathbb{E}_{\hat{w}\sim p_G}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w}) \\
&\leq \mathbb{E}_{c,w}\, F_D(\mathbb{E}_{\hat{w}\sim p_G}\mathbf{1}_{\hat{w}=w}) \\
&\leq \mathbb{E}_{c,w}\, F_D(p_G(w|c)) \\
&= \mathbb{E}_{c,w}\, F_D(e^{\log P_G(w|c)}) \\
&\leq F_D(\mathbb{E}_{c,w} e^{\log P_G(w|c)}) \\
&= F_D\left(\mathbb{E}_{c,w}\left[e^{\mathbb{E}\log p_G(w|c)} + e^{\mathbb{E}\log p_G(w|c)}(\log p_G(w|c) - \mathbb{E}\log p_G(w|c)) + \right.\right. \\
&\qquad\qquad \left.\left. \frac{1}{2}e^{\mathbb{E}\log p_G(w|c)+\varepsilon}(\log p_G(w|c) - \mathbb{E}\log p_G(w|c))^2\right]\right) \\
&= F_D\left(e^{\mathbb{E}\log p_G(w|c)} + \frac{V_G}{2}e^{\mathbb{E}\log p_G(w|c)+\varepsilon}\right) \\
&= F_D\left(\left(1+\frac{V_G}{2}\right)e^{-L(\theta_G)} + \frac{V_G}{2}e^{\varepsilon}\right),
\end{aligned}
$$

where $V_G = \mathbb{E}(\log p_G(w|c) - \mathbb{E}\log p_G(w|c))^2$.

□

Theorem 1.

Proof. This is a direct consequence of Lemma 1 and Lemma 2. We can write

$$
\begin{aligned}
L^*(\theta_D) \leq \tilde{L}^*(\theta_D) &= \bar{L}_G(\theta_D) + (\tilde{L}^*(\theta_D) - \bar{L}_G(\theta_D)) \\
&\leq F_D\left((1+V_G/2)\, e^{-\bar{L}(\theta_G)} + V_G/2\; e^{\varepsilon}\right) + 2^{-1/2} m_D L(\theta_G)^{1/2}.
\end{aligned}
\tag{A.3}
$$

Note that the discriminator optimization only applies on the first term. Using the existing result Proposition 1 on both the generator and discriminator optimization yields the theorem.

□

A.2 Hyperparameter settings

Our hyperparameter settings follow the standard practice in previous works. For MLM pretraining of the generator, we fix the mask ratio as 15%. When sampling sequences for pretraining, we respect document boundaries and avoid concatenating texts from different documents. We did not mask special tokens following the standard BERT practice. We conduct pretraining on NVIDIA Tesla V100 with 32GB memory and fine-tuning on NVIDIA Tesla P100 with 16GB memory. Table A.1 lists the detailed hyperparameters used in pretraining. Table A.2 lists the detailed hyperparameters used for fine-tuning.

Table A.1. Hyperparameter settings used in pretraining.

Hyperparameters	**Base**	**Large**
Max Steps	125K	125K
Optimizer	Adam	Adam
Peak Learning Rate (Generator)	2×10^{-4}	2×10^{-4}
Peak Learning Rate (Discriminator)	1.5×10^{-3}	1.5×10^{-3}
Batch Size	2048	2048
Warm-Up Steps	10K	10K
Sequence Length	512	512
Relative Position Encoding Buckets	32	128
Relative Position Encoding Max Distance	128	256
Adam ε	1e$-$6	1e$-$6
Adam (β_1, β_2)	$(0.9, 0.98)$	$(0.9, 0.98)$
Clip Norm	2.0	2.0
Dropout	0.1	0.1
Weight Decay	0.01	0.01

A.3 A Roadmap to Hyperparameter Tuning

The separate generator and discriminator optimizers introduced by DecoupledOptim are in line with our understanding of the distinct roles of generator optimization and discriminator optimization in ELECTRA-style training. Therefore, DecoupledOptim is much more friendly to hyperparameter tuning, which often requires excessive efforts especially for LLM pre-training.

Table A.2. Hyperparameter search space in fine-tuning.

Hyperparameters	**Base**	**Large**
Sequence Length	256	256
Optimizer	AdaMax	AdaMax
Peak Learning Rate	{5e-5,1e-4, 3e-4}	{5e-5,1e-4, 3e-4}
Max Epochs	{2,3,5,10, 20}	{2,3,5,10, 20}
Batch size	{16, 32, 64, 128}	{8, 16, 32, 64}
Learning rate decay	Linear	Linear
Weight Decay	{0, 0.01}	{0, 0.01}
Warm-up Proportion	{6 %, 10 %, 30%}	{6 %, 10 %, 30%}
Adam ε	1e-6	1e-6
Adam (β_1, β_2)	$(0.9, 0.98)$	$(0.9, 0.98)$
Gradient Clipping	1.0	1.0
Dropout	0.1	0.1

Here we provide a guideline on the hyperparameter selection in DecoupledOptim to achieve the best pretraining performance.

First, since DecoupledOptim is no longer sensitive to the generator size, one can choose a generator as large as possible given hardware constraints. Next, one can find a discriminator learning rate as large as possible with the generator learning rate fixed as any value, provided that the training is stable. This is based on our understanding that a larger discriminator learning rate can almost always benefit the pretraining. Finally, one can locate the best generator learning rate through an efficient binary search, since increasing the generator learning rate first helps and then hurts the training. In Figure A.1, we show our practice of finding the best hyperparameter combination for the Base setting.

A.4 Understand Generator "Overfitting" in ELECTRA

In ELECTRA-style pretraining, it has been widely observed that a large generator consistently hurts the discriminator performance, as also shown in Figure 1.1 for "Baseline". It has been speculated that a large generator may prevent the discriminator from learning effectively (Clark et al., 2020), yet the exact reason remains largely unclear.

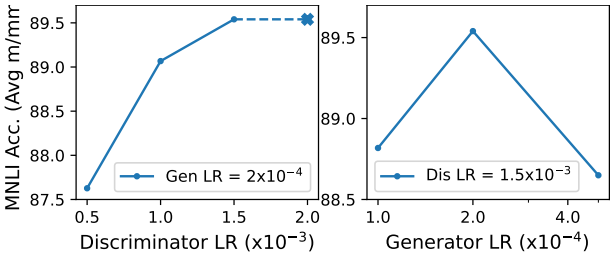


Figure A.1. Practice of finding the best hyperparameter combination for the Base setting. (Left): We search the largest possible discriminator learning rate as long as the training is stable. (Right): Upon locate the largest discriminator learning rate, we binary search the best generator learning rate since the its correlation with the performance is an unimodal curve.

Does a large generator hurt cross-domain generalization? A natural explanation to the "overfitting" phenomenon in ELECTRA-style pretraining is that a large generator may impair the cross-domain generalization, namely, it hurts the transferability of the discriminator to downstream tasks since the gap between pretraining performance and downstream performance is commonly seen, especially when the pretraining task and the downstream task are significantly different (Zoph et al., 2020; Wei et al., 2021).

However, we show that the generator "overfitting" phenomenon is not, or at least not completely, due to a larger transferring gap between RTD-based pretraining and downstream tasks such as MNLI. A large generator may already hurt the pretraining performance, in that the discriminator becomes less effective on detecting replaced tokens.

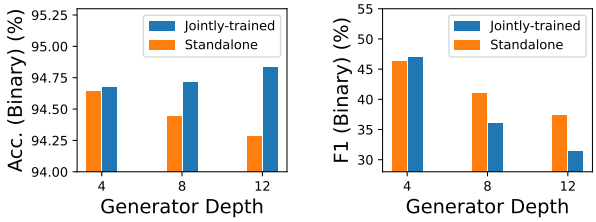


Figure A.2. The binary classification performance of the discriminators on detecting replaced tokens generated by their individual jointly-trained generators versus that by a standalone generator. (Left): Accuracy (Right): F1-score. Here the standalone generator has 4 layers and is trained for 125k steps.

Fair evaluation of RTD performance. To demonstrate this, we need to first measure the

RTD performance of discriminators jointly trained with different generator sizes in a fair manner. The standard binary classification performance reported in RTD-based pertraining is clearly not a measure that is comparable across different discriminators, as they are evaluated with different jointly-trained generators.

Alternatively, we propose to use a standalone generator to measure the RTD performance fairly. A standalone generator is pretrained and shared across evaluations which means we can maintain the difficulty of the generated replaced tokens for different discriminators. As shown in Figure A.2, the binary classification performance of discriminators against such a standalone generator ranks consistently with different metrics, while the jointly-trained generators report mixed rankings. This suggests the standalone generator is more reliable to measure the RTD performance.

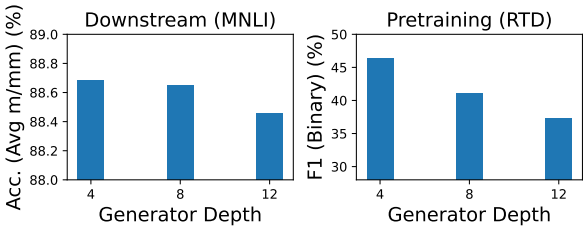


Figure A.3. (Left): Averaged matched/mismatched accuracy on the MNLI dataset by fine-tuning discriminators trained along with different generator sizes. (Right): Fair measure of the RTD performance of discriminators jointly trained with generators of different sizes. Here the standalone generator has 4 layers and is trained for 125k steps.

Large generators cause inferior RTD performance. With a fair measure of the RTD pretraining performance, we can now observe that a larger generator in fact hurts the pretraining. As shown in Figure A.3, when trained with a larger generator, the discriminator is not able to detect the replaced tokens as effectively. Subsequently, the performance on downstream tasks degrades as well. Therefore, to understand and potentially and rectify generator "overfitting", it is necessary to first dig into the pretraining stage of ELECTRA.

A.4.1 Does a large generator hurt in-domain generalization?

We conjecture that a large generator hurts RTD-based pretraining because the replaced tokens generated by it may lack diversity and be overfitted easily, thus hurting RTD performance.

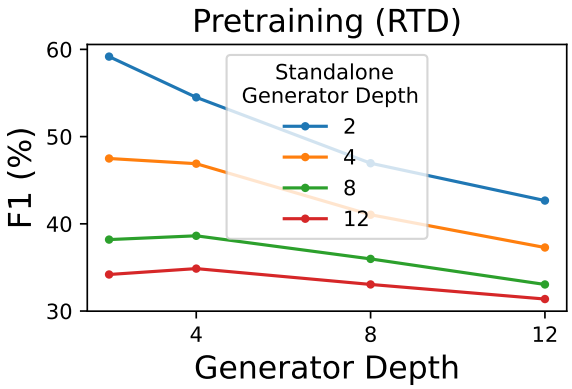


Figure A.4. RTD performance (F1-score) of discriminators jointly trained with generators of different sizes (2, 4, 8, 12 layers), measured against standalone generators of different sizes (2, 4, 8, 12 layers)

Overfitting the jointly-trained generator? One possibility is that a large generator may predict the token at some masked positions with a low-entropy distribution, which means the replaced token at this particular position may almost always be the same. This prevents the discriminator from learning to detect other diverse replaced tokens. However, we find that this might not be the case. For example, a BERT-Base discriminator gets an RTD performance of 31.4% in terms of F1-score against its jointly-trained generator with 12 layers. In comparison, when evaluated against a standalone generator sharing the exact same architecture as the jointly-trained generator, but initialized with a different random seed, it gets an F1-score of 30.9%, which shows no significant discrepancy. Therefore, it is not likely that the discriminator is overfitting a jointly-trained large generator itself.

Overfitting the size? It is also possible that large generators may generate replaced tokens with similar properties. For example, the replaced tokens generated by different generators, albeit randomly initialized, are the same or follow similar distributions. In this case, the discriminator may overfit generators of this particular size. However, Figure A.4 shows that this may not be

the case. A large generator also hurts the RTD performance against a standalone generator of exactly the same size. Furthermore, one may find that a larger generator in fact hurts the RTD performance against standalone generators of multiple different sizes, which suggests that a large generator may hurt RTD-based pretraining "universally".

A.4.2 How to reliably measure the performance of RTD pretraining?

In the above sections we mentioned that to reliably measure the performance of RTD pretraining, it is better to evaluate the discriminator against a standalone generator. However, we note that not all standalone generators can reliably measure the RTD performance such that it can reflect the downstream performance. As shown in Figure A.4, a standalone generator with 2 layers will erroneously report the discriminator trained with a 2-layer generator is better than the one trained with a 4-layer generator, while on downstream tasks the latter is better. Therefore, we speculate that this is because the replaced token distributions generated by a 2-layer generator is too simple in the verge of random, thus is not able to reflect the discriminator's capacity of language understanding. Based on this observation, we believe to reliably measure the RTD performance we should sample replaced tokens from a distribution that is sufficiently difficult, for example, a generator with a reasonably large size.

Appendix B

Model-Generated Data from "Supportive" Models

B.1 Proof

B.1.1 Modified Softmax

We provide more details for the modified Softmax we defined. Recall that

Definition 6 (Modified Softmax)**.**

$$\widetilde{Softmax}(\hat{h}) = \frac{\exp(\frac{1}{M}\sum_m \hat{h}_m)}{(\prod_m \sum_k \exp(\hat{h}_{m,k}))^{1/M}}. \tag{B.1}$$

Our modified Softmax layer maps $\hat{h} \in \mathbb{R}^{M\times K}$ to the probabilistic output $f(x) \in \mathbb{R}^K$. It can be viewed as a combination of an average pooling layer and a Softmax layer, where the denominator of the softmax layer is a geometric average of the sum-exp of the probabilistic output of each batch $\hat{h}_m$.

B.1.2 Lemma 3

Proof. Given the labeled dataset $\mathscr{D} = \{(x^{(i)}, y^{(i)})\}$, the empirical risk of the network function f under cross-entropy loss can be written as

$$
\begin{aligned}
& -\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log f(x^{(i)}) \\
=& -\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log \widetilde{\mathrm{Softmax}}(w_C f_E(x^{(i)})) \\
:=& -\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log \widetilde{\mathrm{Softmax}}\left(\hat{h}^{(i)}\right) \\
=& -\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log \frac{\exp(\frac{1}{M}\sum_m \hat{h}_m^{(i)})}{(\prod_m \sum_k \exp(\hat{h}_{m,k}^{(i)}))^{1/M}} \\
=& -\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log \left(\frac{\prod_m \exp \hat{h}_m^{(i)})}{\prod_m \sum_k \exp(\hat{h}_{m,k}^{(i)})} \right)^{1/M} \\
=& -\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \log \frac{\prod_m \exp(\hat{h}_m^{(i)})}{\prod_m \sum_k \exp(\hat{h}_{m,k}^{(i)})} \\
=& -\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_m \log \frac{\exp(\hat{h}_m^{(i)})}{\sum_k \exp(\hat{h}_{m,k}^{(i)})} \\
=& -\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_m \log \mathrm{Softmax}(\hat{h}_m^{(i)}) \\
:=& -\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_m \log p_m^{(i)},
\end{aligned}
\tag{B.2}
$$

where we have defined $p_m^{(i)} = \mathrm{Softmax}(\hat{h}_m^{(i)}) \equiv \mathrm{Softmax}(w_c f_E(x_m^{(i)}))$. Here $x_m^{(i)}$ indicates the m-th patch of the input x.

Now recall our assumption that f_E is an invariant feature extractor, which means that for any i and j, any m and m', and any γ and γ', $f_E(\gamma(x_m^{(i)})) = f_E(\gamma'(x_m^{(j)}))$ as long as $z_m = z_{m'}$. Since, $p_m^{(i)} = \mathrm{Softmax}(\hat{h}_m^{(i)}) = \mathrm{Softmax}(w_C f_E(\gamma(x_m^{(i)})))$, this means that $p_m^{(i)} = p_{m'}^{(j)}$ as long as $z_m = z_{m'}$.

Therefore, the empirical risk can be regrouped as

$$\begin{aligned}
&-\frac{1}{MN}\sum_i 1(y^{(i)})\cdot\sum_m \log p_m^{(i)} \\
=&-\frac{1}{MN}\sum_i 1(y^{(i)})\cdot\sum_{z\in\mathscr{Z}^i}\log p_z \\
=&-\frac{1}{M}\sum_{z\in\mathscr{Z}}\left(\frac{1}{N}\sum_{\{i|z\in\mathscr{Z}^{(i)}\}}1(y^{(i)})\right)\cdot\log p_z,
\end{aligned}$$

where we have let $p_z := p_m^{(i)}$ where $z_m = z$.

Now we use the KKT condition to approach the empirical risk minimization problem since the only variable is p_z. The first-order stationarity condition gives

$$\nabla_{p_z}\cdot = -\frac{1}{MN}\sum_i 1(y^{(i)})\odot\frac{1}{p_z}+\lambda\mathbf{1}=0, \tag{B.3}$$

where $\odot$ indicates the Hadamard product. Now Hadamard product both sides by p_z we have

$$-\frac{1}{MN}\sum_i 1(y^{(i)})+\lambda p_z = 0,$$

and thus

$$p_z = \frac{1}{MN\lambda}\sum_i 1(y^{(i)}).$$

Now consider the condition $\mathbf{1}\cdot p_z = 1$, we have

$$\frac{1}{MN\lambda}\sum_i 1(y^{(i)})\cdot\mathbf{1} = 1.$$

Since $\sum_i 1(y^{(i)})\cdot\mathbf{1} = |\{i|z\in\mathscr{Z}^{(i)}\}|$,

$$\lambda = \frac{|\{i|z\in\mathscr{Z}^{(i)}\}|}{MN}.$$

This leads to

$$p_z = \frac{1}{N_z} \sum_{\{i|z \in \mathscr{Z}^{(i)}\}} 1(y^{(i)}),$$

where $N_z \coloneqq |\{i|z \in \mathscr{Z}^{(i)}\}|$.

□

B.1.3 Lemma 4

Proof. The sum of independently but non-identically distributed multinomial random variables is known as the Possion multinomial distribution (PMD). We utilize a known result of PMD to prove this lemma.

Proposition 2 (Lin et al. (2022))**.** *Let $\boldsymbol{I}_i = (I_{i1,\cdots,I_{im}}), i = 1,\cdots,n$ be n independent random indicators where $I_{ij} \in \{0,1\}$ and $\sum_{j=1}^{m} I_{ij} = 1$ for each i. Let $\boldsymbol{p}_i = (p_{i1},\cdots,p_{im})$ be the probability vector that $\boldsymbol{I}_i$ is sampled from, where $\sum_{j=1}^{m} p_{ij} = 1$. Let $\boldsymbol{X}$ be the sum of these n random indicators, namely $\boldsymbol{X} = (X_1,\cdots,X_m) = \sum_{i=1}^{n} \boldsymbol{I}_i$. Then we have*

$$\mathbb{E}[\boldsymbol{X}] = (p_{\cdot 1},\cdots,p_{\cdot m}), \tag{B.4}$$

where $p_{\cdot j} = \sum_{i=1}^{n} p_{ij}$.

And

$$\Sigma_{jk} = \begin{cases} \sum_{i=1}^{n} p_{ij}(1-p_{ij}), \text{ if } j = k, \\ -\sum_{i=1}^{n} p_{ij}p_{ik}, \text{ if } j \neq k, \end{cases} \tag{B.5}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of $\boldsymbol{X}$.

Following this result, since $1(y^{(i)})$ is a random indicator and $y^{(i)} \sim p_Y(\cdot|x^{(i)})$, we have

$$\mathbb{E}[\bar{y}_z] = \bar{p}(\cdot|z) \coloneqq \frac{1}{N} \sum_{\{i|z \in \mathscr{Z}^{(i)}\}} p_Y(\cdot|x^{(i)}). \tag{B.6}$$

And let the covariance matrix of $\bar{y}_z$ to be $\Sigma_{\bar{y}_z}$. Let $N_z := |\{i|z \in \mathscr{Z}^{(i)}\}|$ be the number of inputs that contain feature z, we have

$$\begin{aligned}
\mathrm{tr}[\Sigma_{\bar{y}_z}] &= \frac{1}{N^2} \sum_{\{i|z\in\mathscr{Z}^{(i)}\}} \sum_{k=1}^{K} p_Y(k|x^{(i)})(1-p_Y(k|x^{(i)})) \\
&= \frac{1}{N^2} \sum_{\{i|z\in\mathscr{Z}^{(i)}\}} \sum_{k=1}^{K} \left(1-p_Y^2(k|x^{(i)})\right) \\
&\leq \frac{(K-1)N_z}{N^2},
\end{aligned} \tag{B.7}$$

where we utilized the fact that $\sum_k p_Y^2(k|x^{(i)}) \geq \frac{1}{K}$ by the Cauchy-Schwarz inequality.

Then by a multivariate Chebyshev inequality (see, *e.g.*, Chen (2007)), we have with probability $1-\delta$,

$$\|\bar{y}_z - \bar{p}(\cdot|z)\|_2^2 \leq \frac{\mathrm{tr}(\Sigma_{\bar{y}_z})}{\delta} \leq \frac{(K-1)N_z}{N^2} \cdot \frac{1}{\delta}. \tag{B.8}$$

Now finally, we derive an estimation of N_z. Recall that each input contains M feature names sampled from the feature vocabulary $|\mathscr{Z}|$ subject to the distribution p_Z. Therefore,

$$\mathbb{E}\left[\frac{N_z}{N}\right] = P(z \in \mathscr{Z}^{(i)}) = 1-(1-p_Z(z))^M \approx M p_Z(z). \tag{B.9}$$

For simplicity, we assume p_Z is a uniform distribution, which means $p_Z(z) \approx 1/|\mathscr{Z}|$, for any z. Note that this is not a necessary assumption since N_z can also be trivially bounded by N. Therefore we have

$$\|\bar{y}_z - \bar{p}(\cdot|z)\|_2^2 \leq \frac{(K-1)M}{N|\mathscr{Z}|} \cdot \frac{1}{\delta}. \tag{B.10}$$

Recall that M is the number of features (patches) in each input, and $|\mathscr{Z}|$ is the total number of features in the vocabulary. This result thus indicates a larger feature vocabulary can improve the approximation while a more complicated input can hurt the approximation.

□

B.1.4 Lemma 5

Proof. To prove this lemma we need an additional weak assumption on the true label distribution of features that are "similar".

Assumption 2. *Let $I(z,z')$ define the pointwise mutual information (PMI) between features z and z'. Let ψ be a concave and monotonically* ***decreasing*** *function. We assume*

$$D_{KL}\left(p_Y(\cdot|z)\,\middle\|\,p_Y(\cdot|z')\right) = \psi\left(I(z,z')\right). \tag{B.11}$$

Eq. (B.11) indicates that when sampling the features in an input, if two features are more likely to be sampled together, their true label distribution should be more similar.

Given Eq. (B.11) and recall that

$$p_Y(\cdot|x^{(i)}) = \left(\prod_m p_Y(\cdot|z_m)\right)^{1/M} = \left(\prod_{z\in\mathscr{Z}^{(i)}} p_Y(\cdot|z)\right)^{1/M},$$

it can be shown that the difference between the true label distribution of the input that contains feature z and the true label distribution of feature z can be bounded. In specific, for $i \in \{i'|z \in$

$\mathscr{Z}^{(i')}\}$, we have

$$
\begin{aligned}
D_{\mathrm{KL}}\left(p_Y(\cdot|z)\,\middle\|\,p_Y(\cdot|x^{(i)})\right) &= -\sum_k p_Y(k|z)\log\left(\frac{p_Y(k|x^{(i)})}{p_Y(k|z)}\right) \\
&= -\frac{1}{M}\sum_k p_Y(k|z)\log\left(\prod_{z'\in\mathscr{Z}^{(i)}}\frac{p_Y(k|z')}{p_Y(k|z)}\right) \\
&= -\frac{1}{M}\sum_k p_Y(k|z)\sum_{z'\in\mathscr{Z}^{(i)}}\log\left(\frac{p_Y(k|z')}{p_Y(k|z)}\right) \\
&= -\frac{1}{M}\sum_{z'\in\mathscr{Z}^{(i)}}\sum_k p_Y(k|z)\log\left(\frac{p_Y(k|z')}{p_Y(k|z)}\right) \\
&= \frac{1}{M}\sum_{z'\in\mathscr{Z}^{(i)}} D_{\mathrm{KL}}\left(p_Y(\cdot|z)\,\middle\|\,p_Y(\cdot|z')\right) \\
&= \frac{1}{M}\sum_{z'\in\mathscr{Z}^{(i)}}\psi\left(I(z,z')\right), \\
&\leq \psi(\bar{I}(i,z)).
\end{aligned}
\tag{B.12}
$$

where $\bar{I}(i,z) = \frac{1}{M}\sum_{z'\in\mathscr{Z}^{(i)}} I(z,z')$ is the average PMI between feature z and other features z' in an input that contains z. Here we utilize Jensen's inequality on concave functions.

Now we can show that the average true label distribution of inputs that contain feature z approximates the true label distribution of feature z. Recall the definition of average true label

distribution of inputs that contain feature z is $\bar{p}(\cdot|z) \coloneqq \frac{1}{N}\sum_{\{i|z\in\mathscr{Z}^{(i)}\}} p_Y(\cdot|x^{(i)})$. And we have

$$
\begin{aligned}
\|\bar{p}(\cdot|z) - p_Y(\cdot|z)\|_1 &= \left\| \left(\frac{1}{N} \sum_{\{i|z\in\mathscr{Z}^{(i)}\}} p_Y(\cdot|x^{(i)}) \right) - p_Y(\cdot|z) \right\|_1 \\
&\leq \frac{1}{N} \left\| \sum_{\{i|z\in\mathscr{Z}^{(i)}\}} \left(p_Y(\cdot|x^{(i)}) - p_Y(\cdot|z) \right) \right\|_1 \\
&\leq \frac{1}{N} \sum_{\{i|z\in\mathscr{Z}^{(i)}\}} \left\| p_Y(\cdot|x^{(i)}) - p_Y(\cdot|z) \right\|_1 \\
&\leq \frac{2^{1/2}}{N} \sum_{\{i|z\in\mathscr{Z}^{(i)}\}} D_{\mathrm{KL}}^{1/2} \left(p_Y(\cdot|z) \| \, p_Y(\cdot|x^{(i)}) \right) \\
&\leq \frac{2^{1/2}}{N} \sum_{\{i|z\in\mathscr{Z}^{(i)}\}} \psi^{1/2}(\bar{I}(i,z)) \\
&\leq \frac{2^{1/2} N_z}{N} \psi^{1/2}\left(\bar{I}(z)\right) \\
&\approx \frac{2^{1/2} M}{|\mathscr{Z}|} \psi^{1/2}\left(\bar{I}(z)\right),
\end{aligned}
\tag{B.13}
$$

where $\bar{I}(z) = \frac{1}{N_z} \min_{\{i|z\in\mathscr{Z}^{(i)}\}} \bar{I}(i,z)$ is the average PMI over all inputs that contain feature z.

□

B.1.5 Theorem 2

Proof. Recall the probabilistic output of the minimizer of the empirical risk is

$$
f^*(x^{(i)}) = \left(\prod_m p_m^{(i)} \right)^{1/M} = \left(\prod_{z\in\mathscr{Z}^{(i)}} p_z \right)^{1/M}, \tag{B.14}
$$

where p_z defined in Section B.1.2 is the network's probabilistic output of each feature z.

Note that for all k, $f(x^{(i)})(k) \leq 1$ and $p_Y(k|x^{(i)}) \leq 1$. Therefore,

$$
\begin{aligned}
\left\| f^*(x^{(i)}) - p_Y(\cdot|x^{(i)}) \right\|_2 &\leq \exp\left(\left\| \log(f^*(x^{(i)})) - \log(p_Y(\cdot|x^{(i)})) \right\|_2 \right) \\
&= \exp\left(\frac{1}{M} \left\| \sum_{z \in \mathscr{Z}^{(i)}} (\log p_z - \log p_Y(\cdot|z)) \right\|_2 \right) \\
&\leq \exp\left(\frac{1}{M} \sum_{z \in \mathscr{Z}^{(i)}} \| \log p_z - \log p_Y(\cdot|z) \|_2 \right) \\
&\leq \exp\left(\frac{1}{M} \sum_{z \in \mathscr{Z}^{(i)}} \beta \log \| p_z - p_Y(\cdot|z) \|_2 \right) \\
&\leq \left(\max_{z \in \mathscr{Z}^{(i)}} \| p_z - p_Y(\cdot|z) \|_2 \right)^{\beta} \\
&= \left(\max_{z \in \mathscr{Z}^{(i)}} \| \bar{y}_z - p_Y(\cdot|z) \|_2 \right)^{\beta} \\
&= \left(\max_{z \in \mathscr{Z}^{(i)}} \left(\| \bar{y}_z - \bar{p}(\cdot|z) \|_2 + \| \bar{p}(\cdot|z) - p_Y(\cdot|z) \|_2 \right) \right)^{\beta} \\
&\leq \left(\max_{z \in \mathscr{Z}^{(i)}} \left(\| \bar{y}_z - \bar{p}(\cdot|z) \|_2 + \| \bar{p}(\cdot|z) - p_Y(\cdot|z) \|_1 \right) \right)^{\beta} \\
&= \left(\sqrt{\frac{(K-1)M}{N|\mathscr{Z}|} \frac{1}{\delta}} + \frac{2^{1/2} M}{|\mathscr{Z}|} \psi^{1/2}(\bar{I}_{\min}) \right)^{\beta},
\end{aligned}
\tag{B.15}
$$

where $\beta \coloneqq \min(\min_k f(x^{(i)})(k), \min_k p_Y(k|x^{(i)}))$ and $\bar{I}_{\min} = \min_{z \in \mathscr{Z}^{(i)}} \bar{I}(z)$.

□

B.1.6 Theorem 3

Proof. We first present a Lemma similar to Lemma 3, which shows that the probabilistic predictions of features will still converge to the sample mean of the labels where the corresponding inputs contain this feature, up to some constant error.

Lemma 6 (Convergence of the probabilistic predictions of features with Lipschitz-continuous and transformation-robust feature extractor)**.** *Let* $\bar{y}_z \coloneqq \frac{1}{N} \sum_{\{i|z \in \mathscr{Z}^{(i)}\}} 1(y^{(i)})$*, where* $\mathscr{Z}^{(i)}$ *denote the set of feature names in the i-th input, and thus* $\{i|z \in \mathscr{Z}^{(i)}\}$ *denotes the set of inputs that contain*

feature z. Let f^* be a minimizer of the empirical risk (Eq. (2.1)) and assume f_E^* is a L_X-Lipschitz-continuous and L_Γ-transformation-robust feature extractor. Let $p_{f^*}(x_z) \coloneqq Softmax(w_C f_E^*(x_z))$. We have with probability $1-\delta$,

$$\|p_{f^*}(x_z) - \bar{y}_z\| \leq L_z, \tag{B.16}$$

where $L_z \coloneqq 2L_\Gamma + L_X \tilde{O}(\frac{\nu_z}{\delta})$.

Proof. Since f_E^* is a L_X-Lipschitz-continuous and L_Γ-transformation-robust feature extractor, for any i, j, for any m, m' and any γ, γ', as long as $z_m = z_{m'} \equiv z$, we will have ,

$$\begin{aligned}
&\|f_E(x_m^{(i)}) - f_E(x_{m'}^{(j)})\| \\
=&\|f_E(\gamma(x_z^{(i)})) - f_E(\gamma'(x_z^{(j)}))\| \\
\leq&\|f_E(\gamma(x_z^{(i)})) - f_E(x_z^{(i)})\| + \|f_E(\gamma'(x_z^{(j)})) - f_E(x_z^{(j)})\| + \|f_E(x_z^{(i)}) - f_E(x_z^{(j)})\| \\
\leq&2L_\Gamma + L_X\|x_z^{(i)} - x_z^{(j)}\|
\end{aligned} \tag{B.17}$$

Now since the feature representation in the input space is always concentrated, by Chebyshev's inequality we will have with probability $1-\delta$

$$\|x_z^{(i)} - x_z^{(j)}\| \leq \tilde{O}\left(\frac{\nu_z}{\delta}\right).$$

Therefore, we have with probability $1-\delta$,

$$\|f_E(x_m^{(i)}) - f_E(x_{m'}^{(j)})\| \leq L_z,$$

where $L_z \coloneqq 2L_\Gamma + L_X \tilde{O}\left(\frac{\nu_z}{\delta}\right)$.

The empirical loss minimization now becomes

$$\begin{aligned}
&\min_f -\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_m \log p_m^{(i)} \\
s.t. \quad &\|p_m^{(i)} - p_{m'}^{(j)}\| \leq L_z, \text{ if } z_m = z_{m'}.
\end{aligned} \tag{B.18}$$

Let $\mathscr{S}_z \coloneqq \{i | z \in \mathscr{Z}^{(i)}\}$ for simplicity, we can have

$$\begin{aligned} -\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_m \log p_m^{(i)} &= -\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_{z \in \mathscr{Z}^{(i)}} \log p_z^{(i)} \\ &= -\frac{1}{MN}\sum_{z \in \mathscr{Z}} \sum_{\{i | z \in \mathscr{Z}^{(i)}\}} 1(y^{(i)}) \cdot \log p_z^{(i)}, \\ &= -\frac{1}{MN}\sum_{z \in \mathscr{Z}} \sum_{i \in \mathscr{S}_z} 1(y^{(i)}) \cdot \log p_z^{(i)}, \end{aligned} \tag{B.19}$$

Note that this basically means that we assign the label of an example $y^{(i)}$ to each feature z contained in this example's input.

Therefore, (B.18) will be equivalent to the following problem.

$$\begin{aligned} &\min_{\{p_z^{(i)}\}} -\frac{1}{MN}\sum_{z \in \mathscr{Z}} \sum_{i \in \mathscr{S}_z} 1(y^{(i)}) \cdot \log p_z^{(i)} \\ &s.t. \quad \|p_z^{(i)} - p_z^{(j)}\| \leq L_z, \mathbf{1} \cdot p_z^{(i)} = 1, \ \forall\, z, \ \forall\, i, j \in \mathscr{S}_z, i \neq j. \end{aligned}$$

Note that the constraint $\|\cdot\| \leq L_z$ is only imposed in each subset $\mathscr{S}_z$.

Using the KKT condition, the above problem can be further formulated as follows.

$$\begin{aligned} &\min_{\{p_z^{(i)}\}} -\frac{1}{MN}\sum_{z \in \mathscr{Z}} \sum_{i \in \mathscr{S}_z} 1(y^{(i)}) \cdot \log p_z^{(i)} \\ &\quad + \sum_z \sum_{j,k \in \mathscr{S}_z, j \neq k} \mu_{z;jk} \left(\|p_z^{(j)} - p_z^{(k)}\| - L_z \right) \\ &\quad + \sum_z \sum_{l \in \mathscr{S}_z} \lambda_{z;l} (\mathbf{1} \cdot p_z^{(l)} - 1), \\ &s.t. \quad \mu_{z;jk}(\|p_z^{(j)} - p_z^{(k)}\| - L_z) = 0, \ \ \mu_{z;jk} \geq 0, \ \forall\, z, \ \forall\, j,k \in \mathscr{S}_z, j \neq k, \\ &\quad \|p_z^{(j)} - p_z^{(k)}\| \leq L_z, \ \ \mathbf{1} \cdot p_z^{(l)} = 1, \ \forall\, z, \ \forall\, j,k,l \in \mathscr{S}_z, j \neq k. \end{aligned} \tag{B.20}$$

Using the first-order stationarity condition we have, $\forall\, z, \ \forall\, i \in \mathscr{S}_z$,

$$\nabla_{p_z^{(i)}} \cdot = -\frac{1}{MN} 1(y^{(i)}) \odot \frac{1}{p_z^{(i)}} + 2\sum_{k \neq i} \mu_{ik}(p_z^{(i)} - p_z^{(k)}) + \lambda_i \mathbf{1} = 0, \tag{B.21}$$

where for simplicity we neglected the subscript z since the condition is the same for all z.

Sum (B.21) over i we have

$$-\frac{1}{MN}\sum_i 1(y^{(i)}) \odot \frac{1}{p_z^{(i)}} + 2\sum_i \sum_{k\neq i} \mu_{ik}(p_z^{(i)} - p_z^{(k)}) + \sum_i \lambda_i \mathbf{1} = 0. \tag{B.22}$$

Note that $\sum_i \sum_{k\neq i} \mu_{ik}(p_z^{(i)} - p_z^{(k)}) = 0$ since each pair of i,k appears twice in the sum, where μ_{ik} is the same but the sign of $p_z^{(i)} - p_z^{(k)}$ is different. Therefore

$$-\frac{1}{MN}\sum_i 1(y^{(i)}) \odot \frac{1}{p_z^{(i)}} + \sum_i \lambda_i \mathbf{1} = 0. \tag{B.23}$$

Notice that to minimize the loss, if $y^{(i)} = y^{(j)}$, it is necessary that $p_z^{(i)} = p_z^{(j)}$.

Now we dot product both sides of (B.23) with $1(k)$, we have

$$-\frac{1}{MN}\frac{|\{i|y^{(i)} = k\}|}{p_z^{(i)}[k]} + \sum_i \lambda_i = 0, \ \forall\, i \in \{i|y^{(i)} = k\},$$

which means that

$$p_z^{(i)}[k] = \frac{|\{i|y^{(i)} = k\}|}{MN\sum_i \lambda_i}, \text{ if } y^{(i)} = k.$$

Now recall the constraint that when $i \neq j$,

$$\|p_z^{(i)} - p_z^{(j)}\| \leq L_z.$$

This at least indicates that

$$\|p_z^{(i)} - p_z^{(j)}\|_\infty \leq L_z.$$

Therefore,

$$p_z^{(i)}[k'] - \frac{|\{i|y^{(i)} = k'\}|}{MN\sum_i \lambda_i} \leq L_z, \text{ if } k' \neq y^{(i)}.$$

This implies that

$$\left\| p_z^{(i)} - \frac{1}{MN\sum_i \lambda_i} \sum_i 1(y^{(i)}) \right\| \leq L_z$$

Now consider the condition $\mathbf{1} \cdot p_z^{(i)} = 1$, we will have

$$\sum_i \lambda_i = \frac{N_z}{MN}.$$

Thus

$$\left\| p_z^{(i)} - \bar{y}_z \right\| \leq L_z,$$

where we recall $\bar{y}_z = \frac{1}{N_z} \sum_i 1(y^{(i)})$. □

Now to prove Theorem 3, we only need to combine Lemma 6 and Lemmas 4, 5, where the reasoning is exactly same as the proof of Theorem 2.

□

B.2 Limitations

In this paper we focus on the theoretical feasibility of learning the true label distribution of training examples with empirical risk minimization. Therefore we only analyze the existence of such a desired minimizer, but neglect the optimization process to achieve it. By explore the optimization towards true label distribution, potentially more dynamics can be found to inspire new regularization techniques.

Also, our proposed method for training a student-oriented teacher may not be able to advance the state-of-the-art significantly, as the regularization techniques based inspired by our analyses (e.g., Lipschitz regularization and consistency regularization) may more or less be leveraged by existing training practice of deep neural networks, either implicitly or explicitly.

B.3 Implementation details

Lipschitz regularization. Following previous practice using Lipschitz regularization for generalization on unseen data (Yoshida & Miyato, 2017) or stabilizing generative model (Miyato et al., 2018), we regularize the Lipschitz constant of a network by constraining the Lipschitz constant of each trainable component. The regularization term is thus defined as $\ell_{\mathrm{LR}} = \sum_f \mathrm{Lip}(f)$, where f denotes a trainable component in the network $\boldsymbol{f}$. The Lipschitz constant of a network component $\mathrm{Lip}(f)$ induced by a norm $\|\cdot\|$ is the smallest value L such that for any input features h, h', $\|f(h) - f(h')\| \leq L\|h - h'\|$. Here we adopt the Lipschitz constant induced by 1-norm, since its calculation is accurate, simple and efficient. For calculating the Lipschitz constants of common trainable components in deep neural networks, we refer to (Gouk et al., 2021) for a comprehensive study.

Consistency regularization. We design our consistency regularization term as $\ell_{\mathrm{CR}} = \frac{1}{N}\sum_i \left\| \boldsymbol{f}(x_i) - \overline{\boldsymbol{f}(x_i)} \right\|_2^2$, where we follow previous work (Laine & Aila, 2017) and employ MSE to penalize the difference. Here $\overline{\boldsymbol{f}(x)}$ is the aggregated prediction of an input x, which we calculate as the simple average of previous predictions $\overline{\boldsymbol{f}(x)}_t = \frac{1}{t}\sum_{t'=0}^{t-1} \boldsymbol{f}(x)_{t'}$, where we omit the data augmentation operator for simplicity. At epoch 0 we simply skip the consistency regularization. Note that such a prediction average can be implemented in an online manner thus there is no need to store every previous prediction of an input.

B.4 Details of experiment setting

B.4.1 Hyperparameter setting for teacher network training

For all the experiments on CIFAR-100, we employ SGD as the optimizer and train for 240 epochs with a batch size of 64. The learning rate is initialized at 0.05 and decayed by a factor of 10 at the epochs 150, 180 and 210, with an exception for ShuffleNet where the learning rate is initialized at 0.01 following existing practice (Tian et al., 2020; Park et al., 2021). The weight decay and momentum are fixed as 0.0005 and 0.9 respectively. The training images are

augmented with random cropping and random horizontal flipping with a probability of 0.5.

For Tiny-ImageNet experiments, we employ SGD as the optimizer and conduct the teacher training for 90 epochs with a batch size of 128. The learning rate starts at 0.1 and is decayed by a factor of 10 at epochs 30 and 60. The weight decay and momentum are fixed as 0.0005 and 0.9 respectively. The training images are augmented with random rotation with a maximum degree of 20, random cropping and random horizontal flipping with a probability of 0.5. For student training the only difference is that we train for additional 10 epochs, with one more learning rate decay at epoch 90, aligned with previous settings (Tian et al., 2020).

For consistency regularization in our teacher training method, we experiment with various weight schedules besides the linear schedule mentioned in the main paper. We list the formulas for these schedules in the following. Here t denotes the epoch number, T denotes the total number of epochs, and $\lambda_{CR}^{\max}$ denotes the maximum weight.

- Cosine schedule:

$$\lambda_{CR}(t) = \cos\left[\left(1-\frac{t}{T}\right)\frac{\pi}{2}\right]\lambda_{CR}^{\max}$$

- Cyclic schedule:

$$\lambda_{CR}(t) = \sqrt{1-\left(1-\frac{t}{T}\right)^2}\,\lambda_{CR}^{\max}$$

- Piecewise schedule:

$$\lambda_{CR}(t) = \begin{cases} 0, & 0 < t \le T/3, \\ \lambda_{CR}^{\max}/2, & T/3 < t \le 2T/3, \\ \lambda_{CR}^{\max}, & 2T/3 < t \le T. \end{cases}$$

B.4.2 Hyperparameter setting for knowledge distillation algorithms

For knowledge distillation algorithms we refer to the setting in RepDistiller [1]. Specifically, for original KD, the loss function used for student training is defined as

$$\ell = \alpha \ell_{\text{Cross-Entropy}} + (1-\alpha)\ell_{KD}.$$

We grid search the best hyper-parameters that achieve the optimal performance, namely the loss scaling ratio α is set as 0.5 and the temperature is set as 4 for both CIFAR-100 and Tiny-ImageNet. For all feature distillation methods combined with KD the loss function can be summarized as (Tian et al., 2020)

$$\ell = \gamma \ell_{\text{Cross-Entropy}} + \alpha \ell_{KD} + \beta \ell_{\text{Distill}},$$

where we grid search the optimal γ and α to be 1.0 and 1.0 respectively. When using our teacher training method, all these hyperparameters are kept same except that for all feature distillation algorithms the scaling weights corresponding to the feature distillation losses β are cut by half, as we wish to rely more on the original KD that is well supported by our theoretical understanding. Table B.1 list β used in our experiments for all feature distillation algorithms. For SSKD (Xu et al., 2020a) the hyperparameters are set as $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 2.7$, $\lambda_4 = 10.0$ for standard training and $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 1.0$, $\lambda_4 = 10.0$ for our methods. For the curriculum distillation algorithm RCO we experiment based on one-stage EEI (equal epoch interval). We select 24 anchor points (or equivalently every 10 epochs) from the teacher's saved checkpoints.

B.5 Additional experiment results

Training overhead. Compared to standard teacher training, the computation overhead of DecoupledOptimis mainly due to the calculation of the Lipschitz constant, which is efficient as it

[1] https://github.com/HobbitLong/RepDistiller

Table B.1. β for different feature distillation algorithms

	β			β			β	
	Standard	SoTeacher		Standard	SoTeacher		Standard	SoTeacher
FitNets	100	50	AT	1000	500	SP	3000	1500
CC	0.02	0.01	VID	1.0	0.5	RKD	1.0	0.5
PKT	30000	15000	AB	1.0	0.5	FT	200	100
NST	50	25	CRD	0.8	0.5			

Table B.2. DecoupledOptimconsistently outperforms Standard on CIFAR-100 with various KD algorithms.

	WRN40-2/WRN40-1		WRN40-2/WRN16-2		ResNet32x4/ShuffleV2	
	Standard	DecoupledOptim	Standard	DecoupledOptim	Standard	DecoupledOptim
FitNet	74.06 ± 0.20	**74.88** ± 0.15	75.42 ± 0.38	**75.64** ± 0.20	76.56 ± 0.15	**77.91** ± 0.21
AT	73.78 ± 0.40	**75.12** ± 0.17	75.45 ± 0.28	**75.88** ± 0.09	76.20 ± 0.16	**77.93** ± 0.15
SP	73.54 ± 0.20	**74.71** ± 0.19	74.67 ± 0.37	**75.94** ± 0.20	75.94 ± 0.16	**78.06** ± 0.34
CC	73.46 ± 0.12	**74.76** ± 0.16	75.08 ± 0.07	**75.67** ± 0.39	75.43 ± 0.19	**77.68** ± 0.28
VID	73.88 ± 0.30	**74.89** ± 0.19	75.11 ± 0.07	**75.71** ± 0.19	75.95 ± 0.11	**77.57** ± 0.16
RKD	73.41 ± 0.47	**74.66** ± 0.08	75.16 ± 0.21	**75.59** ± 0.18	75.28 ± 0.11	**77.46** ± 0.10
PKT	74.14 ± 0.43	**74.89** ± 0.16	75.45 ± 0.09	**75.53** ± 0.09	75.72 ± 0.18	**77.84** ± 0.03
AB	73.93 ± 0.35	**74.86** ± 0.10	70.09 ± 0.66	**70.38** ± 0.87	76.27 ± 0.26	**78.05** ± 0.21
FT	73.80 ± 0.15	**74.75** ± 0.13	75.19 ± 0.15	**75.68** ± 0.28	76.42 ± 0.17	**77.56** ± 0.15
NST	73.95 ± 0.41	**74.74** ± 0.14	74.95 ± 0.23	**75.68** ± 0.16	76.07 ± 0.08	**77.71** ± 0.10
CRD	74.44 ± 0.11	**75.06** ± 0.37	75.52 ± 0.12	**75.95** ± 0.02	76.28 ± 0.13	**78.09** ± 0.13
SSKD	75.82 ± 0.22	**75.94** ± 0.18	76.31 ± 0.07	**76.32** ± 0.09	78.49 ± 0.10	**79.37** ± 0.11
RCO	74.50 ± 0.32	**74.81** ± 0.04	75.24 ± 0.34	**75.50** ± 0.12	76.75 ± 0.13	**77.59** ± 0.31

only requires simple arithmetic calculations of the trainable weights of a neural network (see Section 2.4). Empirically we observe that training with DecoupledOptimis only slightly longer than the standard training for about 5%. The memory overhead of DecoupledOptimis incurred by buffering an average prediction for each input. However, since such prediction requires no gradient calculation we can simply store it in a memory-mapped file.

B.6 Experiments with uncertainty regularization methods on unseen data

Uncertainty learning on unseen data. Since the objective of our student-oriented teacher training is to learn label distributions of the training data, it is related to those methods aiming to learn quality uncertainty on the unseen data. We consider those methods that are feasible for

Table B.3. Performance of the knowledge distillation when training the teacher using existing regularization methods for learning quality uncertainty on unseen data.

	WRN40-2/WRN40-1	
	Student	Teacher
Standard	73.73 ± 0.13	76.38 ± 0.13
DecoupledOptim	$\mathbf{74.35} \pm 0.23$	74.95 ± 0.28
ℓ_2 (5×10^{-4})	73.73 ± 0.13	76.38 ± 0.13
ℓ_1 (10^{-5})	73.60 ± 0.15	73.52 ± 0.05
Mixup ($\alpha = 0.2$)	73.19 ± 0.21	77.30 ± 0.20
Cutmix ($\alpha = 0.2$)	73.61 ± 0.26	78.42 ± 0.07
Augmix ($\alpha = 1, k = 3$)	73.83 ± 0.09	77.80 ± 0.30
CRL ($\lambda = 1$)	74.13 ± 0.29	76.69 ± 0.16

large teacher network training, including (1) classic approaches to overcome overfitting such as ℓ_1 and ℓ_2 regularization, (2) modern regularizations such as label smoothing (Szegedy et al., 2016) and data augmentations such as mixup (Zhang et al., 2018a) and Augmix (Hendrycks et al., 2020), (3) Post-training methods such as temperature scaling (Guo et al., 2017a), as well as (4) methods that incorporate uncertainty as a learning ojective such as confidence-aware learning (CRL) (Moon et al., 2020).

We have conducted experiments on CIFAR-100 using all these methods and the results can be found in Appendix B.6. Unfortunately, the performance of these regularization methods is unsatisfactory in knowledge distillation — only CRL can slightly outperform the standard training. We believe the reasons might be two-folds. First, most existing criteria for uncertainty quality on the unseen data such as calibration error (Naeini et al., 2015) or ranking error (Geifman et al., 2019), only require the model to output an uncertainty estimate that is correlated with the probability of prediction errors. Such criteria may not be translated into the approximation error to the true label distribution. Second, even if a model learns true label distribution on unseen data, it does not necessarily have to learn true label distribution on the training data, as deep neural networks tend to memorize the training data.

Experiment setup. We conduct experiments on CIFAR-100 with teacher-student pair WRN40-2/WRN40-1. We employ the original KD as the distillation algorithm. The hyperparameter

settings are the same as those mentioned in the main results (see Appendix B.4). For each regularization method we grid search the hyperparameter that yields the best student performance. The results are summarized in Table B.3.

Classic regularization. We observe that with stronger ℓ_2 or ℓ_1 regularization the student performance will not deteriorate significantly as teacher converges. However, it also greatly reduces the performance of the teacher. Subsequently the performance of the student is not improved as shown in Table B.3.

Label smoothing. Label smoothing is shown to not only improve the performance but also the uncertainty estimates of deep neural networks (Müller et al., 2019). However, existing works have already shown that label smoothing can hurt the effectiveness of knowledge distillation (Müller et al., 2019), thus we neglect the results here. An intuitive explanation is that label smoothing encourages the representations of samples to lie in equally separated clusters, thus "erasing" the information encoding possible secondary classes in a sample (Müller et al., 2019).

Data augmentation. Previous works have demonstrated that mixup-like data augmentation techniques can greatly improve the uncertainty estimation on unseen data (Thulasidasan et al., 2019; Hendrycks et al., 2020). For example, Mixup augmented the training samples as $x := \alpha x + (1-\alpha)x'$, and $y := \alpha y + (1-\alpha)y'$, where (x', y') is a randomly drawn pair not necessarily belonging to the same class as x.

As shown in Table B.3, stronger mixup can improve the performance of the teacher, whereas it can barely improve or even hurt the performance of the student. Based on our theoretical understanding of knowledge distillation, we conjecture the reason might be that mixup distorts the true label distribution of an input stochastically throughout the training, thus hampering the learning of true label distribution.

Temperature scaling. Previous works have suggested using the uncertainty on a validation set to tune the temperature for knowledge distillation either in standard learning (Menon et al., 2021) or robust learning (Dong et al., 2021). However, the optimal temperature may not be well aligned with that selected based on uncertainty (Menon et al., 2021). We neglect the experiment

results here as the distillation temperature in our experiments is already fine-tuned.

Uncertainty learning. CRL designs the loss function as $\ell = \ell_{\text{CE}} + \lambda \ell_{\text{CRL}}$, where ℓ_{CE} is the cross-entropy loss and ℓ_{CRL} is an additional regularization term bearing the form of

$$\ell_{\text{CRL}} = \max\left(0, -g(c(x_i), c(x_j))(p(x_i) - p(x_j)) + |c(x_i) - c(x_j)|\right), \tag{B.24}$$

where $p(x) = \max_k \mathbf{f}(x)^k$ is the maximum probability of model's prediction on a training sample x and

$$c(x) = \frac{1}{t-1} \sum_{t'=1}^{t-1} 1(\arg\max_k \mathbf{f}(x)_{t'}^k = y)$$

is the frequency of correct predictions through the training up to the current epoch. Here $g(c_i, c_j) = 1$ if $c_i > c_j$ and $g(c_i, c_j) = -1$ otherwise. Although originally proposed to improve the uncertainty quality of deep neural networks in terms of ranking, we found that CRL with a proper hyperparameter can improve the distillation performance, as shown in Table B.3.

We note that the effectiveness of CRL on distillation can be interpreted by our theoretical understanding, as its regularization term (B.24) is essentially a special form of consistency regularization. To see this we first notice (B.24) is a margin loss penalizing the difference between $p(x)$ and $c(x)$ in terms of ranking. We then rewrite $c(x)$ as

$$c(x) = \mathbf{1}_y \cdot \frac{1}{t-1} \sum_{t'=1}^{t-1} \text{Onehot}[\mathbf{f}(x)_{t'}], \tag{B.25}$$

which is similar to our consistency regularization target, except the prediction is first converted into one-hot encoding. Such variation may not be the best for knowledge distillation as we wish those secondary class probabilities in the prediction be aligned as well.

Appendix C

Rule-Generated Data

C.1 Appendix

Effect of selection ratio. As mentioned before, our methods do not introduce additional hyperparameters. Nevertheless, for pseudo-label selection in general, the selection fraction could be an important hyperparameter as the noise rate in the selected subset of pseudo-labels can vary significantly if we select different fractions, as also shown in Figure 3.1. Therefore, we check the performance of the standard confidence-based pseudo-label selection and the confidence-based selection equipped with seed deletion and random deletion, as the selection fraction varies. Figure C.1 shows that our proposed methods are consistently better than standard confidence and achieve relatively robust performance as the selection fraction varies. The performance peaks when the selection fraction is moderate ($\sim 50\%$) for different datasets.

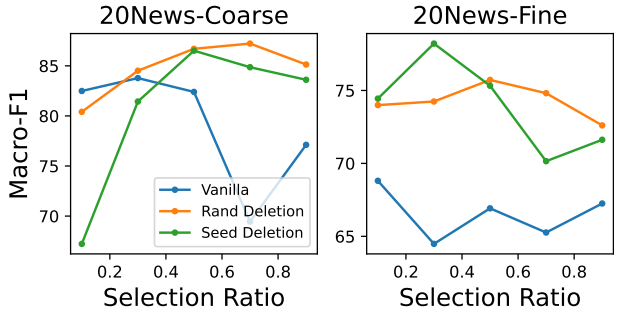


Figure C.1. The classification performance when selecting the pseudo-labels at different fractions.

Experiments on additional confidence regularization methods. We experiment on additional

methods for regularizing the confidence learning of the text classifier, including the following.

- *MC-Dropout* (Gal & Ghahramani, 2015) randomizes the network inference process by dropping intermediate activations. The average output over multiple inferences is utilized as a more reliable confidence score.
- *Early stopping* is often utilized as a regularization to help mitigate overfitting. Empirically, it is observed that an early-stopped model is less prone to learning noisy data (Arpit et al., 2017).

Here, we treat each method as a baseline and compare it with the corresponding method combined with random deletion. For each method, we modulate its most important hyperparameter, namely number of passes for MC-Dropout and number of training epochs for early stopping respectively. As shown in Figures C.2 and C.3, random deletion consistently outperforms the baseline for different confidence regularization methods. Random deletion also achieves more robust performance across different hyperparameter settings, which is important for weakly-supervised classification since we often lack a large clean dataset to select the best hyperparameter.

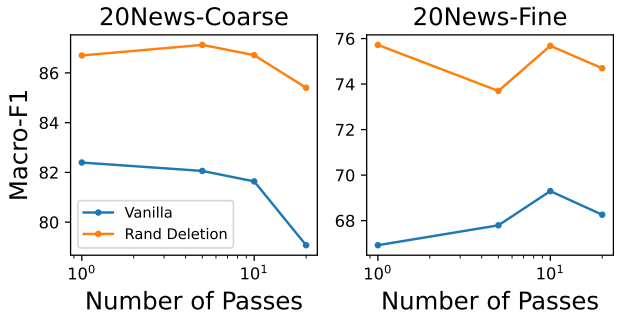


Figure C.2. Classification performance using MC-dropout to obtain better confidence for pseudo-label selection. We test different numbers of passes for MC-dropout.

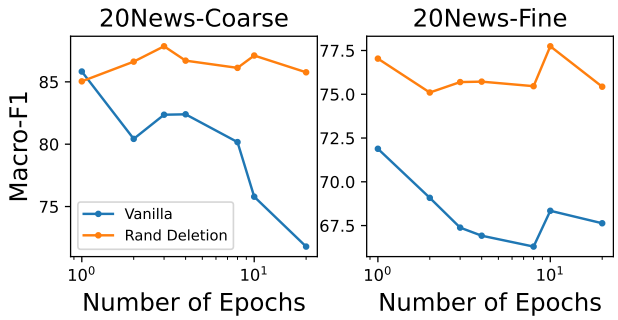


Figure C.3. Classification performance using early stopping as a regularization method to obtain better confidence for pseudo-label selection. We check the performance when early stopping at different epochs.

Bibliography

Tiny imagenet, 2017. URL https://www.kaggle.com/c/tiny-imagenet.

Ahn, S., Hu, S. X., Damianou, A. C., Lawrence, N. D., and Dai, Z. Variational information distillation for knowledge transfer. *CVPR*, pp. 9155–9163, 2019.

Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *ArXiv*, abs/2012.09816, 2020.

Arora, S., Cohen, N., Golowich, N., and Hu, W. A convergence analysis of gradient descent for deep linear neural networks. *ArXiv*, abs/1810.02281, 2018.

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A. C., Bengio, Y., and Lacoste-Julien, S. A closer look at memorization in deep networks. *ArXiv*, abs/1706.05394, 2017.

Bajaj, P., Xiong, C., Ke, G., Liu, X., He, D., Tiwary, S., Liu, T.-Y., Bennett, P., Song, X., and Gao, J. Metro: Efficient denoising pretraining of large scale autoencoding language models with model generated signals. *ArXiv*, abs/2204.06644, 2022.

Bao, H., Dong, L., Wei, F., Wang, W., Yang, N., Liu, X., Wang, Y., Piao, S., Gao, J., Zhou, M., and Hon, H.-W. Unilmv2: Pseudo-masked language models for unified language model pre-training. *ArXiv*, abs/2002.12804, 2020.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. Spectrally-normalized margin bounds for neural networks. In *NIPS*, 2017.

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *ArXiv*, abs/1911.09785, 2019.

Blei, D. M., Ng, A., and Jordan, M. I. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3: 993–1022, 2001.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *Proceedings of the*

12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 535–541, 2006.

Chen, X. A new generalization of chebyshev inequality for random vectors. *ArXiv*, abs/0707.0805, 2007.

Chen, X., Salem, A., Backes, M., Ma, S., and Zhang, Y. Badnl: Backdoor attacks against nlp models. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

Chi, Z., Huang, S., Dong, L., Ma, S., Singhal, S., Bajaj, P., Song, X., and Wei, F. Xlm-e: Cross-lingual language model pre-training via electra. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

Cho, J. H. and Hariharan, B. On the efficacy of knowledge distillation. *ICCV*, pp. 4793–4801, 2019.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. Electra: Pre-training text encoders as discriminators rather than generators. *ArXiv*, abs/2003.10555, 2020.

Dai, J., Chen, C., and Li, Y. A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7:138872–138878, 2019.

Dao, T., Kamath, G. M., Syrgkanis, V., and Mackey, L. W. Knowledge distillation as semiparametric inference. *ArXiv*, abs/2104.09732, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019a.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019b. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Dong, C., Liu, L., and Shang, J. Double descent in adversarial training: An implicit label noise perspective. *ArXiv*, abs/2110.03135, 2021.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A. S., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru,

A., Rozière, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E. A., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I. M., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J. L., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J.-Q., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., neth Heafield, K.-., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M. B., Singh, M., Paluri, M., Kardas, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M. H. M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N. S., Duchenne, O., cCelebi, O., Alrassy, P., Zhang, P., Li, P., Vasić, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S. C., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., ney Meers, W., Martinet, X., Wang, X., Tan, X. E., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A. K., Grattafiori, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Vaughan, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Franco, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, P.-Y. B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Civin, D., Beaty, D., Kreymer, D., Li, S.-W., Wyatt, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Ozgenel, F., Caggioni, F., Guzm'an, F., Kanayet, F. J., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Thattai, G., Herman, G., Sizov, G. G., Zhang, G., Lakshminarayanan, G., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Molybog, I., Tufanov, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres,

J., Ginsburg, J., Wang, J., Wu, K., KamHou, U., Saxena, K., Prasad, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Huang, K., Chawla, K., Lakhotia, K., Huang, K., Chen, L., Garg, L., Lavender, A., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Tsimpoukelli, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Laptev, N. P., Dong, N., Zhang, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollár, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Li, R., Hogan, R., Battey, R., Wang, R., Maheswari, R., Howes, R., Rinott, R., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Shankar, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Cho, S.-B., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Kohler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V. A., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wang, X., Wu, X., Wang, X., Xia, X., Wu, X., Gao, X., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Wang, Y., Hao, Y., Qian, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., and Zhao, Z. The llama 3 herd of models. *ArXiv*, abs/2407.21783, 2024. URL https://api.semanticscholar.org/CorpusID:271571434.

Duerr, S. Seduerr/t5-pawraphrase · hugging face, 2021. URL https://huggingface.co/seduerr/t5-pawraphrase.

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. Born again neural networks. In *ICML*, 2018.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *ArXiv*, abs/1506.02142, 2015.

Gao, T., Yao, X., and Chen, D. Simcse: Simple contrastive learning of sentence embeddings. *ArXiv*, abs/2104.08821, 2021.

Geifman, Y., Uziel, G., and El-Yaniv, R. Bias-reduced uncertainty estimation for deep neural classifiers. In *ICLR*, 2019.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110:393–416, 2021.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks.

ArXiv, abs/1706.04599, 2017a.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017b.

Hao, Y., Dong, L., Bao, H., Xu, K., and Wei, F. Learning to sample replacements for electra pre-training. In *FINDINGS*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CVPR*, pp. 770–778, 2016.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *ArXiv*, abs/2006.03654, 2020.

He, P., Gao, J., and Chen, W. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *ArXiv*, abs/2111.09543, 2021.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. Augmix: A simple data processing method to improve robustness and uncertainty. *ArXiv*, abs/1912.02781, 2020.

Heo, B., Lee, M., Yun, S., and Choi, J. Y. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*, 2019.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, abs/1207.0580, 2012.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models. *ArXiv*, abs/2203.15556, 2022. URL https://api.semanticscholar.org/CorpusID:247778764.

Huang, J., Qu, L., Jia, R., and Zhao, B. O2u-net: A simple noisy label detection approach for deep neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3326–3334, 2019.

Huang, Z. and Wang, N. Like what you like: Knowledge distill via neuron selectivity transfer. *ArXiv*, abs/1707.01219, 2017.

Jin, X., Peng, B., Wu, Y., Liu, Y., Liu, J., Liang, D., Yan, J., and Hu, X. Knowledge distillation

via route constrained optimization. *ICCV*, pp. 1345–1354, 2019.

Kanakarajan, K. R., Kundumani, B., and Sankarasubbu, M. Small-bench nlp: Benchmark for small single gpu trained models in natural language processing. *ArXiv*, abs/2109.10847, 2021.

Ke, G., He, D., and Liu, T.-Y. Rethinking positional encoding in language pre-training. *ArXiv*, abs/2006.15595, 2020.

Kim, J., Park, S., and Kwak, N. Paraphrasing complex network: Network compression via factor transfer. *ArXiv*, abs/1802.04977, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Kurita, K., Michel, P., and Neubig, G. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. *ArXiv*, abs/1610.02242, 2017.

Lin, Z., Wang, Y., and Hong, Y. The poisson multinomial distribution and its applications in voting theory, ecological inference, and machine learning. 2022.

Liu, J., Shou, L., Pei, J., Gong, M., Yang, M., and Jiang, D. Cross-lingual machine reading comprehension with language branch knowledge distillation. In *COLING*, 2020a.

Liu, L., Liu, X., Gao, J., Chen, W., and Han, J. Understanding the difficulty of training transformers. *ArXiv*, abs/2004.08249, 2020b.

Liu, L., Liu, J., and Han, J. Multi-head or single-head? an empirical comparison for transformer training. *ArXiv*, abs/2106.09650, 2021.

Liu, X., He, P., Chen, W., and Gao, J. Multi-task deep neural networks for natural language understanding. In *Annual Meeting of the Association for Computational Linguistics*, 2019a.

Liu, X., Wang, Y., Ji, J., Cheng, H., Zhu, X., Awa, E., He, P., Chen, W., Poon, H., Cao, G., et al. The microsoft toolkit of multi-task deep neural networks for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 118–126, 2020c.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692,

2019b.

Lozhkov, A., Li, R., Allal, L. B., Cassano, F., Lamy-Poirier, J., Tazi, N., Tang, A., Pykhtar, D., Liu, J., Wei, Y., Liu, T., Tian, M., Kocetkov, D., Zucker, A., Belkada, Y., Wang, Z., Liu, Q., Abulkhanov, D., Paul, I., Li, Z., Li, W.-D., Risdal, M. L., Li, J., Zhu, J., Zhuo, T. Y., Zheltonozhskii, E., Dade, N. O. O., Yu, W., Krauss, L., Jain, N., Su, Y., He, X., Dey, M., Abati, E., Chai, Y., Muennighoff, N., Tang, X., Oblokulov, M., Akiki, C., Marone, M., Mou, C., Mishra, M., Gu, A., Hui, B., Dao, T., Zebaze, A. R., Dehaene, O., Patry, N., Xu, C., McAuley, J. J., Hu, H., Scholak, T., Paquet, S., Robinson, J., Anderson, C. J., Chapados, N., Patwary, M., Tajbakhsh, N., Jernite, Y., Ferrandis, C. M., Zhang, L., Hughes, S., Wolf, T., Guha, A., von Werra, L., and de Vries, H. Starcoder 2 and the stack v2: The next generation. *ArXiv*, abs/2402.19173, 2024. URL https://api.semanticscholar.org/CorpusID:268063676.

Matsubara, Y. torchdistill: A modular, configuration-driven framework for knowledge distillation. In *International Workshop on Reproducible Research in Pattern Recognition*, pp. 24–44. Springer, 2021.

Mekala, D. and Shang, J. Contextualized weak supervision for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 323–333, 2020.

Mekala, D., Zhang, X., and Shang, J. Meta: Metadata-empowered weak supervision for text classification. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

Mekala, D., Dong, C., and Shang, J. Lops: Learning order inspired pseudo-label selection for weakly supervised text classification. *ArXiv*, abs/2205.12528, 2022.

Meng, Y., Shen, J., Zhang, C., and Han, J. Weakly-supervised neural text classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 983–992. ACM, 2018.

Meng, Y., Zhang, Y., Huang, J., Xiong, C., Ji, H., Zhang, C., and Han, J. Text classification using label names only: A language model self-training approach. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

Meng, Y., Xiong, C., Bajaj, P., Tiwary, S., Bennett, P., Han, J., and Song, X. Coco-lm: Correcting and contrasting text sequences for language model pretraining. *ArXiv*, abs/2102.08473, 2021.

Meng, Y., Xiong, C., Bajaj, P., Tiwary, S., Bennett, P., Han, J., and Song, X. Pretraining text encoders with adversarial mixture of training signal generators. *ArXiv*, abs/2204.03243, 2022.

Menon, A. K., Rawat, A. S., Reddi, S. J., Kim, S., and Kumar, S. A statistical perspective on distillation. In *ICML*, 2021.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *ArXiv*, abs/1802.05957, 2018.

Mobahi, H., Farajtabar, M., and Bartlett, P. Self-distillation amplifies regularization in hilbert space. *Advances in Neural Information Processing Systems*, 33:3351–3361, 2020.

Moon, J., Kim, J., Shin, Y., and Hwang, S. Confidence-aware learning for deep neural networks. In *ICML*, 2020.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? In *NeurIPS*, 2019.

Naeini, M. P., Cooper, G. F., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. *Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, 2015:2901–2907, 2015.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In *North American Chapter of the Association for Computational Linguistics*, 2019.

Pang, B. and Lee, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 2005.

Park, D. Y., Cha, M., Jeong, C., Kim, D., and Han, B. Learning student-friendly teacher networks for knowledge distillation. *ArXiv*, abs/2102.07650, 2021.

Park, S. and Lee, J. Lime: Weakly-supervised text classification without seeds. *ArXiv*, abs/2210.06720, 2022.

Park, W., Kim, D., Lu, Y., and Cho, M. Relational knowledge distillation. *CVPR*, pp. 3962–3971, 2019.

Passalis, N. and Tefas, A. Learning deep representations with probabilistic knowledge transfer. In *ECCV*, 2018.

Peng, B., Jin, X., Liu, J., Zhou, S., Wu, Y., Liu, Y., Li, D., and Zhang, Z. Correlation congruence for knowledge distillation. *ICCV*, pp. 5006–5015, 2019.

Raffel, C., Shazeer, N. M., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683, 2019.

Rizve, M. N., Duarte, K., Rawat, Y. S., and Shah, M. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. *ArXiv*, abs/2101.06329, 2021.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2015.

Shah, H., Khare, A., Shah, N., and Siddiqui, K. Kd-lib: A pytorch library for knowledge distillation, pruning and quantization, 2020.

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *ArXiv*, abs/2001.07685, 2020.

Stanton, S., Izmailov, P., Kirichenko, P., Alemi, A. A., and Wilson, A. G. Does knowledge distillation really work? *Advances in Neural Information Processing Systems*, 34:6906–6919, 2021.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. *CVPR*, pp. 2818–2826, 2016.

Tan, M., Chen, B., Pang, R., Vasudevan, V., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. *CVPR*, pp. 2815–2823, 2019.

Tang, J. and Wang, K. Ranking distillation: Learning compact ranking models with high performance for recommender system. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

Tang, J., Shivanna, R., Zhao, Z., Lin, D., Singh, A., Chi, E. H., and Jain, S. Understanding and improving knowledge distillation. *ArXiv*, abs/2002.03532, 2020.

Tao, F., Zhang, C., Chen, X., Jiang, M., Hanratty, T., Kaplan, L., and Han, J. Doc2cube: Automated document allocation to text cube via dimension-aware joint embedding. *Dimension*, 2016:2017, 2015.

Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T., and Michalak, S. E. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *NeurIPS*, 2019.

Tian, Y., Krishnan, D., and Isola, P. Contrastive representation distillation. *ArXiv*, abs/1910.10699, 2020.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. Efficient object localization using convolutional networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 648–656, 2014.

Tung, F. and Mori, G. Similarity-preserving knowledge distillation. *ICCV*, pp. 1365–1374, 2019.

Villalobos, P., Sevilla, J., Heim, L., Besiroglu, T., Hobbhahn, M., and Ho, A. C. Will we run out of data? limits of llm scaling based on human-generated data. 2022. URL https://api.semanticscholar.org/CorpusID:253397775.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*, 2018.

Wang, Q., Yang, L., Kanagal, B., Sanghai, S. K., Sivakumar, D., Shu, B., Yu, Z., and Elsas, J. L. Learning to extract attribute value from product via question answering: A multi-task approach. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Wang, Z., Mekala, D., and Shang, J. X-class: Text classification with extremely weak supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3043–3053, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.242. URL https://aclanthology.org/2021.naacl-main.242.

Wei, F., Gao, Y., Wu, Z., Hu, H., and Lin, S. Aligning pretraining for detection via object-level contrastive learning. In *Neural Information Processing Systems*, 2021.

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Chapter of the Association for Computational Linguistics*, 2017.

Xie, Q., Dai, Z., Hovy, E. H., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation for consistency training. *arXiv: Learning*, 2020.

Xu, G., Liu, Z., Li, X., and Loy, C. C. Knowledge distillation meets self-supervision. In *ECCV*, 2020a.

Xu, Z., Gong, L., Ke, G., He, D., Zheng, S., Wang, L., Bian, J., and Liu, T.-Y. Mc-bert: Efficient language pre-training via a meta controller. *ArXiv*, abs/2006.05744, 2020b.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*, 2019.

Yang, Z., Shou, L., Gong, M., Lin, W., and Jiang, D. Model compression with two-stage multi-teacher knowledge distillation for web question answering system. *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020.

Yoshida, Y. and Miyato, T. Spectral norm regularization for improving the generalizability of

deep learning. *ArXiv*, abs/1705.10941, 2017.

Yuan, L., Tay, F. E. H., Li, G., Wang, T., and Feng, J. Revisiting knowledge distillation via label smoothing regularization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3902–3910, 2020.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *ArXiv*, abs/1605.07146, 2016.

Zagoruyko, S. and Komodakis, N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *ArXiv*, abs/1612.03928, 2017.

Zhang, H., Cissé, M., Dauphin, Y., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *ArXiv*, abs/1710.09412, 2018a.

Zhang, L., Ding, J., Xu, Y., Liu, Y., and Zhou, S. Weakly-supervised text classification based on keyword graph. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657, 2015.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CVPR*, pp. 6848–6856, 2018b.

Zhang, Y., Baldridge, J., and He, L. Paws: Paraphrase adversaries from word scrambling. *ArXiv*, abs/1904.01130, 2019.

Zhang, Y., Xu, X., Zhou, H., and Zhang, Y. Distilling structured knowledge into embeddings for explainable and accurate recommendation. *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020.

Zhang, Z. and Sabuncu, M. R. Self-distillation as instance-specific label smoothing. *ArXiv*, abs/2006.05065, 2020.

Zhang, Z., Zhao, H., Utiyama, M., and Sumita, E. Language model pre-training on true negatives. *ArXiv*, abs/2212.00460, 2022.

Zhao, X., Ouyang, S., Yu, Z., Wu, M., and Li, L. Pre-trained language models can be fully zero-shot learners. *arXiv preprint arXiv:2212.06950*, 2022.

Zhu, Y., Kiros, R., Zemel, R. S., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 19–27, 2015.

Zoph, B., Ghiasi, G., Lin, T.-Y., Cui, Y., Liu, H., Cubuk, E. D., and Le, Q. V. Rethinking pre-training and self-training. *ArXiv*, abs/2006.06882, 2020.