FTFT: efficient and robust Fine-Tuning by transFerring Training dynamics

Anonymous ACL submission

Abstract

Despite the massive success of fine-tuning large Pre-trained Language Models (PLMs), they remain susceptible to out-of-distribution and adversarial input. Data Map (DM) is a simple yet effective dual-model approach that enhances the robustness of fine-tuned PLMs. It involves fine-tuning a model on the original training set (i.e. reference model), selecting a subset of important training examples based on the training dynamics of the reference model, and fine-tuning the same model only on these selected examples (i.e. main model). However, this approach requires fine-tuning the same model twice, which is computationally expensive for large models. In this paper, we show that 1) training dynamics are highly transferable across model sizes and pre-training methods, and that 2) main models fine-tuned using DM learn faster than when using conventional Empirical Risk Minimization. Building on these observations, we propose a novel fine-tuning approach based on the DM approach: Fine-Tuning by transFerring Training dynamics (FTFT). Compared with DM, FTFT uses more efficient reference models and fewer training steps. FTFT achieves better generalization robustness than ERM while spending less than half of the training cost.[1]

1 Introduction

Current state-of-the-art performance in Natural Language Processing (NLP) is dominated by large, pretrained language models (PLMs), which are typically fine-tuned for downstream tasks. Scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022) suggest that better downstream performance is achieved with larger pretrained language models. However, fine-tuning large PLMs is also more expensive than fine-tuning small PLMs, in terms of both computational resources and carbon emission (Strubell et al., 2019; Wu et al., 2022).

Moreover, despite impressive progress on regular benchmarks, many studies have shown that fine-tuned PLMs lack robustness against out-of-distribution (OOD) input. For instance, human annotators can easily exploit the weaknesses of fine-tuned PLMs to trick these models to yield incorrect predictions, on tasks such as Natural Language Inference (NLI) (Nie et al., 2020) and Hate Speech Detection (HSD) (Vidgen et al., 2021b).

The problem of robustness can be mitigated using dual-model approaches. With such approaches, first a **reference model** is trained to estimate the importance of each training instance, and then a **main model** is trained based on the outputs of the reference model (Nam et al., 2020; Utama et al., 2020; Sanh et al., 2021; Karimi Mahabadi et al., 2020; Zhang et al., 2022; Liu et al., 2021). In particular, the approach proposed by Swayamdipta et al. (2020) is attractive because it is simple and consistently improves model performance on OOD test datasets. First, a **Data Map (DM)** is constructed, based on the **training dynamics** (i.e. instance prediction probabilities) from an initial fine-tuning run of the reference model on the full dataset. This DM then divides training data into three subsets: ambiguous, hard-to-learn, and easy instances. Finally, the main model is fine-tuned using only either the ambiguous or the hard-to-learn subset. In Swayamdipta et al. (2020) the reference model and the main model are the same PLM (e.g., $\text{DeBERTaV3}_{\text{Large}}$; He et al., 2023). However, a major drawback of this approach is that it improves robustness at the expense of efficiency, because it requires fine-tuning the same model twice.

In this paper, we jointly address the issues of robustness and efficiency without sacrificing the simplicity and OOD performance of the DM approach, by exploiting the **transferability of training dynamics**. We make two key contributions.

First, we study an unexplored question: Are data maps transferable across different model sizes and

[1] Our code will be publicly available on GitHub.

pretraining methods? We focus on the novel setting where data maps are created based on computationally efficient reference models to fine-tune more capable — and often larger — main models. We make three key observations (§4). First, in most cases, *training dynamics are highly transferable* across different model sizes and pretraining methods. For example, one can use DeBERTaV3$_{\text{Large}}$ as the main model, but DeBERTaV3$_{\text{Small}}$ or ELECTRA (Clark et al., 2020) as the reference model. Second, the condition for successful transfer is that reference models should be reasonably strong, in a sense that we also seek to make precise. Crucially, we identify a key property of effective reference models, namely that they typically allocate higher ratios of training instances to the easy subset. This observation can help us inspect whether a reference model would work well without training the main model. Third, the main model learns faster using the DM approach than with ERM, i.e., good performance is achieved using fewer steps.

Second, based on these findings, we propose **Fine-Tuning by transFerring Training dynamics (FTFT, §5)**: an efficient fine-tuning approach that leads to improved out-of-distribution performance. Compared to the DM approach, FTFT is more efficient in two ways. First, FTFT uses more efficient reference models to identify ambiguous training instances. Second, FTFT uses substantially fewer training steps to fine-tune the main model. Experiments on two tasks, NLI and HSD, show that FTFT achieves better performance on out-of-distribution input than ERM, while lowering the training cost by a factor of more than two.

2 Background

Improving Model Robustness Using Dual-Model Approaches Many studies have proposed dual-model approaches to improve model robustness, without the knowledge of instance group information. For example, Nam et al. (2020) first train a reference model using generalized cross-entropy loss. Then, they train a main model while assigning higher weights to training instances that are hard for the reference model. Sanh et al. (2021) use a Product-of-Expert (PoE) approach, by first training a reference model with limited capacity to capture dataset biases, and then training the main model to avoid these biases using PoE loss. Liu et al. (2021) propose the Just-Train-Twice approach (JTT), which involves first training a less capable reference model using heavy regularization and vanilla SGD, and then up-weighing the training instances that the reference model predicts incorrectly when training the main model. The DM approach proposed by Swayamdipta et al. (2020) is based on a similar idea, but instead of relying only on the correctness of the reference model, they use training dynamics to categorize training instances. Furthermore, the main model is only trained on a subset of the data. We discuss this method below.

Data Map Swayamdipta et al. (2020) propose a dual-model approach for improving model robustness. First, a reference model is trained on the full training dataset. Then, a Data Map (DM) is built based on the observed training dynamics, by tracking the prediction probabilities of the true class (p_{true}) of each training instance across epochs. Using the DM, training instances can be categorized into three groups: *ambiguous* (i.e. the standard deviation of p_{true} is in the top $q\%$ of all training instances); *hard-to-learn* (i.e. the mean of p_{true} is at the bottom $q\%$ of all training instances); and *easy* (i.e. neither ambiguous nor hard-to-learn). The threshold $q\%$ is fixed and typically set to 33%. Note that a training instance can be categorized as both hard-to-learn and ambiguous (a low mean but high standard deviation for p_{true}). Finally, the main model is fine-tuned only on the $q\%$ most ambiguous or hard-to-learn data points. Swayamdipta et al. (2020) show that, with a slight loss of In-Distribution (ID) performance, this approach improves model performance on challenging Out-Of-Distribution (OOD) datasets. They also observe that training on ambiguous data leads to better performance than training on hard-to-learn data. We therefore mainly focus on ambiguous data.

Swayamdipta et al. (2020) uses the same PLM as both the reference and the main model. In contrast, Sar-Shalom and Schwartz (2023) recently showed that a DM constructed by ELECTRA$_{\text{Large}}$ can be used to improve the robustness of DeBERTaV3$_{\text{Large}}$. However, instead of using only the ambiguous subset, they added k copies of this subset to the original training set. Moreover, they did not investigate either DM transfer across model sizes and pretraining methods, or how such transferability can be exploited to enhance efficiency.

Model-Based Data Selection/Reweighing Our work is also connected to studies that have proposed to use a reference model to select or reweigh data. Chang et al. (2017) use p_{true} variance and

proximity to the classification threshold from a reference model to reweigh training instances; Toneva et al. (2019) calculate the frequency of forgetting events (i.e. from correct to incorrect prediction), and remove the least forgettable instances; Paul et al. (2021) instead use error vector norm to estimate the contribution of a training instance.

Previous studies have also explored the use of a smaller reference model to improve efficiency. For instance, Coleman et al. (2020) use a small model for active learning and core-set selection. Xie et al. (2023) reweigh domains for language model pretraining, by training a small reference model to estimate the difficulty of each domain.

Building on the previously mentioned lines of research, we propose Fine-Tuning by transFerring Training dynamics (FTFT), which is more efficient than the original DM approach, while retaining the advantage of improved robustness.

3 Experimental Setup

We perform our experiments on two tasks, Natural Language Inference (NLI) and Hate Speech Detection (HSD). As a baseline, we also experiment with a random DM (i.e., randomly selecting $q\%$ of the training data). Unless otherwise specified, we follow Swayamdipta et al. (2020) and set the ratio $q\%$ of all DM methods to 33%.

Data To study the impact on model robustness, for each task, besides the training set, we use an ID validation set and a few challenging OOD test sets.

For NLI, we use the MultiNLI dataset (Williams et al., 2018) as the train and ID validation set, because of its diverse composition, covering 10 genres. As OOD test sets, we use two challenging datasets designed to target weaknesses of models trained on MultiNLI: WANLI (Liu et al., 2022) and AdversarialNLI (Nie et al., 2020), which consists of three rounds of adversarial data collection.

For HSD, we use CAD (Vidgen et al., 2021a) as the training and ID validation set. CAD consists of Reddit posts covering diverse topics and writing styles, annotated based on a fine-grained taxonomy. We frame the task as a binary classification task (i.e. hateful vs. non-hateful). Following Ramponi and Tonelli (2022), we mark identity-related abuse as hateful and other categories as non-hateful. As our OOD test sets, we use DynaHate (Vidgen et al., 2021b), which also contains three rounds of adversarial data collection and their perturbations. While conceptualizations of hate speech tend to vary widely across different datasets, we selected these two because their annotations are based on compatible hate speech definitions and taxonomies.

Models We mainly use two PLMs in our experiments, DeBERTaV3 (He et al., 2023) and ELECTRA (Clark et al., 2020), because they offer strong performance and various model sizes. To study the transferability across different pretraining methods, we also use TinyBERT (Turc et al., 2020), BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) as reference models. Costs for fine-tuning different PLMs are in Appendix A.1.[2]

Training For NLI, we train all models for 60k steps and evaluate every 4k steps. For HSD, we train all models for 6k steps and evaluate every 400 steps. For all models, we use four different random seeds and report their average performance. Full training details (e.g., optimization, software and hardware) are included in Appendix A.2.

4 Transferability of Training Dynamics

In this section, we study the transferability of training dynamics in the DM method, i.e., whether we can use different reference and main models while maintaining the robustness advantage of the main model. Specifically, we study whether training dynamics are transferable across different model sizes (§4.1, e.g., from $\text{DeBERTaV3}_{\text{Small}}$ to $\text{DeBERTaV3}_{\text{Large}}$) and pretraining methods (§4.2, e.g., from $\text{ELECTRA}_{\text{Large}}$ to $\text{DeBERTaV3}_{\text{Large}}$).

We focus on these issues for two reasons. First, transferability across model sizes can help improve the efficiency of constructing a DM, by enabling the use of more efficient reference models. Second, transferability across pretraining methods can help achieve such an efficiency gain, when more efficient versions of the main model pretraining method are either not available, or cannot model the task sufficiently well. Moreover, understanding transferability can shed more light on data importance. If DMs of different reference models consistently identify the same subset of training instances as ambiguous, it suggests that DMs reveal intrinsic data characteristics, rather than characteristics that are solely model-dependent.

Concretely, we define successful transfers as transfers that produce comparable or better OOD

[2] We report PFLOPs rather than GPU hours because we noticed occasional low GPU utilization especially when fine-tuning smaller PLMs.

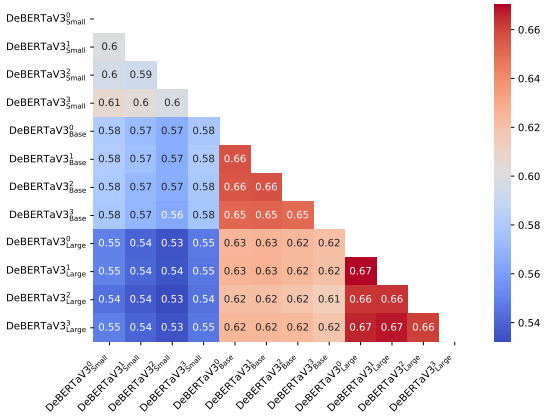

Figure 1: Consistency across different sizes of DeBERTaV3 on NLI. The numbers are the fractions (0–1) of the ambiguous training instances shared by two models. Training dynamics are transferable across different sizes: the fractions of shared ambiguous instances between models of different sizes are only slightly smaller than those between models of the same size but different random seeds (shown as superscript).

performance than training the main model with ERM. Our results show that training dynamics are transferable across different model sizes and pretraining methods, with a few exceptions. To understand the conditions for successful transfers, we analyze these failure cases. Our main finding in this regard (§4.3) is that successful transfer can be achieved with smaller and more efficient reference models as long as they perform reasonably well on identifying easy training instances. This finding can serve as a guideline for choosing reference models without training the main model, which can be computationally expensive.

4.1 Transferability Across Model Sizes

In this section, we study whether smaller and more efficient models can be used as reference models for training larger main models, without compromising robustness. For example, when using DeBERTaV3$_{Large}$ as the main model, we investigate whether there is a comparable performance when using DeBERTaV3$_{Small}$ versus using DeBERTaV3$_{Large}$ itself as reference model. Successful transfers of this type enables the use of more efficient reference models.

We show the results for DeBERTaV3 on NLI and HSD in Table 1 here and Table 5 from Appendix B (across different model sizes), and make three observations. First, larger main models offer better performance. Within the ERM results, there are clear performance differences among different model sizes; also, almost all DM results (which use DeBERTaV3$_{Large}$ as the main model) outperforms ERM with DeBERTaV3$_{Small}$ and DeBERTaV3$_{Base}$. Second, consistent with Swayamdipta et al. (2020), DM and ERM are better at ID and OOD performance respectively. Third and most importantly, *training dynamics are transferable across different model sizes*: When using DeBERTaV3$_{Large}$ as the main model, changing the reference model from DeBERTaV3$_{Large}$ to DeBERTaV3$_{Small}$ or DeBERTaV3$_{Base}$ yields comparable performance. Interestingly, using smaller reference models sometimes results in even better OOD performance: our hypothesis is that these models are more sensitive to data difficulty (e.g., data biases (Sanh et al., 2021; Liu et al., 2021)) compared to larger models; this could be explored further in future work.

To investigate transferability of DMs further, we analyze whether reference models of different sizes identify similar groups of ambiguous instances. Figure 1 shows the fraction of the ambiguous instances shared by different reference models (i.e. of different sizes / random seeds): the fractions of shared ambiguous instances between different sizes are only slightly smaller than those between the same size but different random seeds, providing further evidence for the transferability of DMs.[3]

4.2 Transferability Across Pretraining Methods

We now compare the performance between using reference and main models trained using the same pretraining method and different pretraining methods. If transfers between different pretraining methods are successful, we can still exploit efficient DM-based transfer in case there is no smaller or more efficient version of the main model.

Table 1 here and Table 5 from Appendix B (across different model sizes) show the results for DeBERTaV3$_{Large}$ as the main model with different reference models, on NLI and HSD. *In most settings, training dynamics are transferable across pretraining methods*: DeBERTaV3$_{Large}$ achieves comparable performance using DMs constructed by different reference models in most cases. However, such transfers are not always successful. When using ELECTRA$_{Small}$ as the reference model, the performance is clearly worse on the

[3]The expected fraction that random DMs share is 0.33, since we select the top 33% most ambiguous instances.

Mode	Main Model	Ref. Model	MultiNLI	WANLI	AdversarialNLI R1	AdversarialNLI R2	AdversarialNLI R3
ERM	DeBERTaV3$_{\text{Small}}$	-	$87.57_{0.08}$	$61.61_{0.12}$	$33.55_{1.23}$	$30.63_{0.89}$	$32.40_{0.77}$
ERM	DeBERTaV3$_{\text{Base}}$	-	$90.00_{0.12}$	$64.61_{0.28}$	$43.55_{0.68}$	$33.23_{1.08}$	$34.14_{0.40}$
ERM	DeBERTaV3$_{\text{Large}}$	-	$\mathbf{91.06_{0.08}}$	$66.46_{0.33}$	$58.17_{1.50}$	$45.58_{0.64}$	$41.34_{1.12}$
DM	DeBERTaV3$_{\text{Large}}$	Random	$90.74_{0.14}$	$65.31_{0.78}$	$53.30_{2.33}$	$42.02_{1.36}$	$38.60_{1.17}$
DM	DeBERTaV3$_{\text{Large}}$	DeBERTaV3$_{\text{Large}}$	$90.75_{0.29}$	$66.33_{0.07}$	$59.75_{0.86}$	$45.60_{1.86}$	$41.94_{0.80}$
Across different model sizes							
DM	DeBERTaV3$_{\text{Large}}$	DeBERTaV3$_{\text{Small}}$	$90.74_{0.21}$	$66.80_{0.61}$	$59.60_{1.14}$	$45.62_{1.12}$	$42.04_{0.66}$
DM	DeBERTaV3$_{\text{Large}}$	DeBERTaV3$_{\text{Base}}$	$90.51_{0.05}$	$66.61_{0.76}$	$\mathbf{61.42_{1.44}}$	$46.73_{0.92}$	$41.58_{0.94}$
Across different pretraining methods							
DM	DeBERTaV3$_{\text{Large}}$	ELECTRA$_{\text{Small}}$	$90.91_{0.14}$	$62.09_{0.48}$	$49.63_{0.70}$	$38.50_{1.19}$	$35.98_{0.75}$
DM	DeBERTaV3$_{\text{Large}}$	ELECTRA$_{\text{Base}}$	$90.63_{0.28}$	$66.58_{0.35}$	$59.77_{1.31}$	$46.25_{0.96}$	$42.29_{0.80}$
DM	DeBERTaV3$_{\text{Large}}$	ELECTRA$_{\text{Large}}$	$90.80_{0.20}$	$66.42_{1.07}$	$58.95_{3.12}$	$44.58_{2.11}$	$41.52_{1.46}$
DM	DeBERTaV3$_{\text{Large}}$	BERT$_{\text{Large}}$	$90.03_{0.17}$	$\mathbf{66.89_{0.61}}$	$60.40_{1.25}$	$\mathbf{47.30_{1.15}}$	$\mathbf{43.71_{1.26}}$
DM	DeBERTaV3$_{\text{Large}}$	RoBERTa$_{\text{Large}}$	$90.71_{0.16}$	$66.67_{0.32}$	$58.77_{0.28}$	$46.48_{0.99}$	$41.73_{0.48}$

Table 1: Transferability across different model sizes and pretraining methods, using DeBERTaV3 as the main model on NLI. We compare the performance (accuracy) of 1) DeBERTaV3 of different sizes fine-tuned using ERM, 2) DeBERTaV3$_{\text{Large}}$ as the main model, using a random DM (random 33% training instances), and DeBERTaV3$_{\text{Large}}$ as the reference model (Ref. Model) to construct a DM (original DM), 3) DeBERTaV3$_{\text{Large}}$ as the main model, using DeBERTaV3$_{\text{Small}}$ and DeBERTaV3$_{\text{Base}}$ as reference models to construct a DM, 4) DeBERTaV3$_{\text{Large}}$ as the main model, using ELECTRA of different sizes, BERT$_{\text{Large}}$ and RoBERTa$_{\text{Large}}$, as reference models to construct a DM. R1–R3 in AdversarialNLI refer to different rounds of collected data. Training dynamics are transferrable across different sizes and pretraining methods: Creating DMs using different reference model sizes and pretraining methods results in comparable performance.

NLI OOD datasets than when using ERM. We hypothesize that ELECTRA$_{\text{Small}}$ is not strong enough for constructing effective DMs for MultiNLI (we analyze this further in §4.3 below).

4.3 How Efficient Can We Be?

We have shown that training dynamics are usually transferable across different model sizes and pretraining methods. We now study the conditions for successful transfers, by focusing on two questions: 1) Can we use efficient but weak models as reference models? and 2) What are the differences between effective and ineffective reference models? Answers to these questions can guide the selection of efficient yet effective reference models.

Can we use efficient but weak models as reference models? To answer this question, we compare the performance of a wide range of methods of three types. First, we consider four models fine-tuned with ERM: the small, base, and large versions of ELECTRA, and TinyBERT. We use these four models because they have clearly different sizes and capabilities (ELECTRA$_{\text{Large}}$ $>$ ELECTRA$_{\text{Base}}$ $>$ ELECTRA$_{\text{Small}}$ $>$ TinyBERT). Second, we use these models as reference models to construct DM to fine-tune DeBERTaV3$_{\text{Large}}$. By using reference models with different capabilities to fine-tune the same main model, we can inspect the impact of reference model capability on transferability. Third, we also include the results with ELECTRA$_{\text{Large}}$ as the main model, and different sizes of ELECTRA as reference models. By comparing results with different main models, we can better understand whether successful transfers originate from the compatibility between reference and main models or the capability of the reference model itself. We also include the results for ELECTRA$_{\text{Large}}$ fine-tuned with a random DM as a baseline. We show our results for NLI in Table 2 and include the results for HSD in Table 6 from Appendix B. We observe that both tasks produce consistent results, and make two observations.

First, reference models with poor performance (e.g., TinyBERT and ELECTRA$_{\text{Small}}$) lead to failed transfers. Also, the poorer the performance of the reference models, the worse their main models perform OOD. For instance, TinyBERT reference model results in worse main model performance than ELECTRA$_{\text{Small}}$. Moreover, the success of transfers mostly depends on the reference model, rather than the main model: transfers from ELECTRA$_{\text{Small}}$ to both ELECTRA$_{\text{Large}}$ and DeBERTaV3$_{\text{Large}}$ are unsuccessful.

Mode	Main Model	Ref. Model	MultiNLI	WANLI	AdversarialNLI R1	AdversarialNLI R2	AdversarialNLI R3
ERM	TinyBERT	-	$67.32_{0.19}$	$43.40_{0.16}$	$23.30_{0.22}$	$28.10_{0.67}$	$30.94_{0.27}$
ERM	$\text{ELECTRA}_{\text{Small}}$	-	$81.98_{0.18}$	$54.11_{0.37}$	$23.38_{0.71}$	$28.57_{0.70}$	$30.25_{0.52}$
ERM	$\text{ELECTRA}_{\text{Base}}$	-	$88.53_{0.19}$	$63.06_{0.34}$	$34.58_{1.08}$	$30.73_{0.54}$	$31.29_{1.21}$
ERM	$\text{ELECTRA}_{\text{Large}}$	-	$90.75_{0.19}$	$65.85_{0.41}$	$54.20_{1.13}$	$39.38_{1.17}$	$36.10_{0.38}$
DM	$\text{DeBERTaV3}_{\text{Large}}$	TinyBERT	$89.17_{0.23}$	$60.03_{0.11}$	$41.83_{1.01}$	$34.58_{0.73}$	$34.54_{0.86}$
DM	$\text{DeBERTaV3}_{\text{Large}}$	$\text{ELECTRA}_{\text{Small}}$	$\mathbf{90.91_{0.14}}$	$62.09_{0.48}$	$49.63_{0.70}$	$38.50_{1.19}$	$35.98_{0.75}$
DM	$\text{DeBERTaV3}_{\text{Large}}$	$\text{ELECTRA}_{\text{Base}}$	$90.63_{0.28}$	$\mathbf{66.58_{0.35}}$	$\mathbf{59.77_{1.31}}$	$\mathbf{46.25_{0.96}}$	$\mathbf{42.29_{0.80}}$
DM	$\text{DeBERTaV3}_{\text{Large}}$	$\text{ELECTRA}_{\text{Large}}$	$90.80_{0.20}$	$66.42_{1.07}$	$58.95_{3.12}$	$44.58_{2.11}$	$41.52_{1.46}$
DM	$\text{ELECTRA}_{\text{Large}}$	$\text{ELECTRA}_{\text{Small}}$	$90.40_{0.08}$	$61.53_{0.26}$	$45.90_{1.30}$	$36.20_{1.70}$	$31.89_{1.11}$
DM	$\text{ELECTRA}_{\text{Large}}$	$\text{ELECTRA}_{\text{Base}}$	$89.88_{0.14}$	$66.09_{0.33}$	$54.10_{0.92}$	$40.97_{0.64}$	$37.31_{0.27}$
DM	$\text{ELECTRA}_{\text{Large}}$	$\text{ELECTRA}_{\text{Large}}$	$90.33_{0.11}$	$65.37_{0.61}$	$53.73_{1.29}$	$39.67_{2.08}$	$36.17_{0.22}$
DM	$\text{ELECTRA}_{\text{Large}}$	Random	$89.99_{0.07}$	$65.03_{0.22}$	$51.25_{1.30}$	$39.02_{2.16}$	$34.98_{1.21}$

Table 2: Performance (accuracy) on NLI with conventional fine-tuning (ERM) and DM using the 33% most ambiguous data points identified with different reference models. Random in Ref. Model means randomly selecting 33% of the training data. The rows marked in gray are the results for DM training where the transfer was not successful and the corresponding reference models. Successful transfer requires the reference model to be reasonably strong: reference models with clearly worse performance lead to degraded OOD performance for the main models.

Second, using weak reference models for DM does not negatively affect ID performance much. For instance, transfers from $\text{ELECTRA}_{\text{Small}}$ to $\text{DeBERTaV3}_{\text{Large}}$ yields the best accuracy on MultiNLI. We suspect the reason is that weak models usually identify simple training instances as ambiguous data, and simple training instances have been found to be sufficient for obtaining satisfactory ID performance (Swayamdipta et al., 2020).

What are the differences between effective and ineffective reference models? To answer this question, we consider the differences between a weak and a reasonably strong reference model when categorizing training data into ambiguous, hard-to-learn, and easy subsets. Also, we assume instances in our training set are of different difficulty to the model (i.e. simple to difficult).

Assume we have a weak reference model that can fit simple training instances but cannot fit difficult ones. This weak reference model will therefore assign increasing p_{true} to simple training instances across different epochs, while keeping p_{true} for difficult training instances around the values expected in a random guessing scenario. Thus, p_{true} will exhibit high standard deviations on simpler training instances, which will then be identified as ambiguous data; while more difficult training instances will have both lower means and lower standard deviations for p_{true}, and therefore be identified as hard-to-learn data. In contrast, a reasonably capable reference model can fit simple training instances during the early stage of training, so that

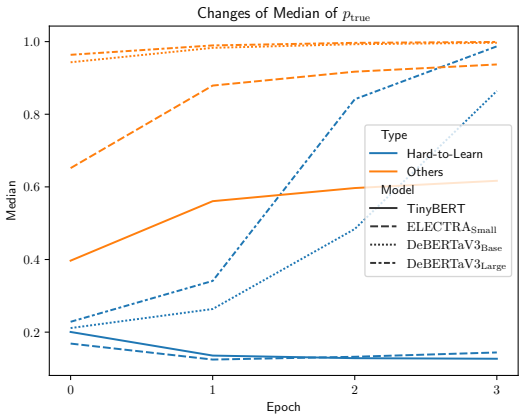

Figure 2: Change of median p_{true} over different training epochs on MultiNLI

these instances have both high means and low standard deviations for p_{true}. In contrast, p_{true} for difficult instances will gradually increase across epochs, making these instances yield relatively low means and high standard deviations for p_{true}. As a result, such instances will be identified as both ambiguous and hard-to-learn (i.e. we expect a large overlap in these subsets). Because we select a fixed percentage $q\%$ of instances as ambiguous or hard-to-learn, the larger overlap results in a larger percentage of data instances to be classified as easy when using strong reference models.

We now show that our reasoning holds in practice. For a given reference model, we first split the training instances into two subsets based on their mean p_{true}: hard-to-learn (10% of all train-

ing instances) and others (the remaining 90%). These two subsets represent more difficult and simpler instances, and we use a lower $q\%$ to make the difference clearer. Then, for each subset, we calculate the median p_{true} in each epoch. We use median values because they are robust statistics of the central tendency. Figure 2 shows our results on MultiNLI. We use two effective (DeBERTaV3$_{\text{Large}}$ and DeBERTaV3$_{\text{Base}}$) and two ineffective (ELECTRA$_{\text{Small}}$ and TinyBERT) reference models. With DeBERTaV3$_{\text{Large}}$ and DeBERTaV3$_{\text{Base}}$, hard-to-learn instances are gradually learned during training, while other instances already have high p_{true} values in the first epoch. In contrast, with ELECTRA$_{\text{Small}}$ and TinyBERT, hard-to-learn data instances are not learned at all, as suggested by their close-to-zero p_{true}, while other instances are gradually fitted by these two models.

To further validate our reasoning, we compute the percentages of training instances identified as easy by different reference models (see the Appendix Table 7 for the full results). Ineffective reference models indeed identify fewer data points as easy compared with other models. For example, on NLI with $q\% = 50\%$, TinyBERT identifies 20.0% of the instances as easy, compared to 46.9% by DeBERTaV3$_{\text{Base}}$. Furthermore, the overlap between hard-to-learn and ambiguous instances in successful transfers is usually very large. For example, with $q\% = 50\%$, all effective reference models identify more than 45% of the training data as easy (the maximum is 50%, when ambiguous and hard-to-learn data align perfectly).

To conclude, reasonably capable reference models that lead to successful transfers 1) fit easier training instances during the early stage of training and 2) fit harder training instances gradually across different epochs; while too weak reference models 1) fit easier training instances gradually across different epochs and 2) fail to fit hard training instances (i.e. p_{true} remains low during training).

5 FTFT: efficient and robust Fine-Tuning by transFerring Training dynamics

Given the transferability of training dynamics across model sizes and pretraining methods (§4), we can improve the efficiency of the DM approach by using more efficient reference models. However, because we need to fine-tune the main model in the same way as ERM, the DM approach is still less efficient. In this section, we address this limitation: we show that with DM we need fewer steps to fine-tune the main model compared to ERM, and propose a novel approach based on this observation: Fine-Tuning by transFerring Training dynamics (FTFT). We show that FTFT consistently offers better efficiency and robustness over ERM.

DM Trains Faster Figure 3 here and Figure 4 from Appendix B show the OOD test performance of ELECTRA$_{\text{Large}}$ fine-tuned with fewer steps (i.e. from 1/15 to 7/15 of the total number of training steps) on NLI and HSD. We compare DM with different reference model sizes against ERM. For each method, we show the relative performance by comparing against a model trained using the total number of training steps. Clearly the DM approach learns much faster than ERM on all OOD datasets: the performance curves of DM methods are almost always above those of ERM. We also observe that DM methods using only 1/3 of the total number of training steps already achieve comparable performance as using all training steps. This result suggests that we can further improve the efficiency of the DM method by training with fewer steps, while maintaining its robustness advantage over ERM. We show the results for DeBERTaV3$_{\text{Large}}$ in Appendix B and observe similar trends.

FTFT: Achieving both Efficiency and Robustness FTFT involves two crucial changes to the DM approach. First, FTFT uses more efficient PLMs as reference models. Second, to improve the efficiency of the main model, FTFT uses only 1/3 of the training steps used for ERM. We choose 1/3 because we select 33% of the most ambiguous training instances, and this choice means we keep the same number of epochs as ERM. Nevertheless, *we recommend to determine the number of training steps by monitoring model performance.* Table 3 summarizes the performance of FTFT using DeBERTaV3$_{\text{Large}}$ as the main model, and DeBERTaV3$_{\text{Small}}$ and DeBERTaV3$_{\text{Base}}$ as reference models. We compare against ERM, original DM, JTT (Liu et al., 2021), and PoE (Sanh et al., 2021). We also show the cost for each method, using the cost of fine-tuning ELECTRA$_{\text{Small}}$ with ERM as unit. The relative cost for each method in HSD is the same as NLI. Note that our cost calculation includes the training of both the reference model and the main model.

We make two observations. First, FTFT achieves better robustness than other methods, indicated by its strong performance on most challeng-

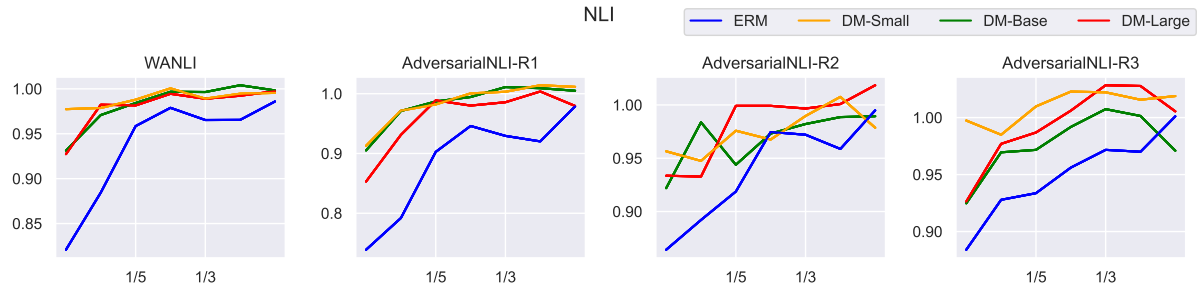

Figure 3: Performance on NLI when training the main model ($\text{ELECTRA}_{\text{Large}}$) with fewer training steps. ERM is standard ERM fine-tuning on the full training set. DM-* refers to fine-tuning $\text{ELECTRA}_{\text{Large}}$ with the DM method, using ELECTRA_{*} as the reference model. The X-axis is the percentile of training steps used, ranging from 1/15 to 7/15 of the total number of training steps. The Y-axis is the **percentage of performance compared with a model trained on the total number of training steps**. Fine-tuning the main model using data maps is much faster than ERM: the models achieve close-to-100% performance with only 1/3 of the training steps.

Mode	Main Model	Ref. Model	Cost	MultiNLI	WANLI	AdversarialNLI R1	AdversarialNLI R2	AdversarialNLI R3
ERM	$\text{DeBERTaV3}_{\text{Large}}$	-	32.0	$\mathbf{91.06_{0.08}}$	$66.46_{0.33}$	$58.17_{1.50}$	$45.58_{0.64}$	$41.34_{1.12}$
DM	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Large}}$	64.0	$90.75_{0.29}$	$66.33_{0.07}$	$59.75_{0.86}$	$45.60_{1.86}$	$41.94_{0.80}$
JTT	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Large}}$	64.0	$90.80_{0.08}$	$66.06_{0.34}$	$59.52_{1.33}$	$45.57_{1.09}$	$41.83_{0.91}$
PoE	$\text{DeBERTaV3}_{\text{Large}}$	TinyBERT	32.0	$91.02_{0.07}$	$\mathbf{67.16_{0.45}}$	$59.80_{1.27}$	$46.45_{0.81}$	$42.71_{0.61}$
FTFT	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Small}}$	15.2	$90.12_{0.77}$	$66.42_{1.26}$	$\mathbf{60.30_{0.94}}$	$45.75_{1.40}$	$\mathbf{43.67_{2.06}}$
FTFT	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Base}}$	19.7	$90.14_{0.41}$	$66.47_{0.79}$	$59.77_{1.76}$	$\mathbf{46.65_{1.86}}$	$42.71_{0.96}$

Mode	Main Model	Ref. Model	Cost	CAD	DynaHate-Original R2	DynaHate-Original R3	DynaHate-Original R4	DynaHate-Perturb R2	DynaHate-Perturb R3	DynaHate-Perturb R4
ERM	$\text{DeBERTaV3}_{\text{Large}}$	-	32.0	$81.69_{0.57}$	$75.44_{1.67}$	$73.32_{0.80}$	$76.12_{1.54}$	$70.62_{1.83}$	$77.41_{0.57}$	$68.89_{1.01}$
DM	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Large}}$	64.0	$81.58_{0.72}$	$79.18_{1.12}$	$76.87_{1.89}$	$77.73_{1.42}$	$73.34_{1.14}$	$76.63_{0.94}$	$67.54_{0.50}$
JTT	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Large}}$	64.0	$81.17_{0.69}$	$75.54_{2.88}$	$72.80_{3.16}$	$76.70_{1.57}$	$72.45_{1.97}$	$76.11_{1.09}$	$67.92_{1.26}$
PoE	$\text{DeBERTaV3}_{\text{Large}}$	TinyBERT	32.0	$\mathbf{81.70_{0.76}}$	$76.87_{1.19}$	$73.48_{1.29}$	$76.58_{1.25}$	$70.46_{1.37}$	$77.17_{0.61}$	$68.28_{1.09}$
FTFT	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Small}}$	15.2	$80.73_{0.77}$	$78.77_{2.39}$	$\mathbf{77.53_{3.11}}$	$\mathbf{79.48_{1.52}}$	$\mathbf{76.19_{1.93}}$	$77.53_{0.82}$	$69.31_{1.69}$
FTFT	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Base}}$	19.7	$79.76_{1.31}$	$\mathbf{82.05_{2.46}}$	$76.77_{2.19}$	$78.62_{1.23}$	$75.22_{2.64}$	$\mathbf{78.43_{1.86}}$	$\mathbf{71.00_{3.15}}$

Table 3: Comparison between FTFT and ERM/original DM fine-tuning. Performance of DeBERTaV3 on NLI (top, accuracy) and HSD (bottom, macro-F1). ERM is conventional ERM fine-tuning, and FTFT refers to using the 33% most ambiguous data identified by different reference models (i.e., Ref. Model) *and only training for 1/3 of the total steps*. Cost refers to the fine-tuning cost, with the cost of fine-tuning ELECTRA-Small with ERM as the unit. FTFT yields both better efficiency and better robustness compared to both ERM fine-tuning and the original DM method.

ing datasets. Second, fine-tuning models using FTFT is also more efficient than other methods. For example, FTFT with $\text{DeBERTaV3}_{\text{Base}}$ and $\text{DeBERTaV3}_{\text{Small}}$ as the reference models only cost 19.7 and 15.2 units of computation: compared with 32(64) units of computation for ERM(DM) fine-tuning, these two FTFT choices are respectively 1.63(3.26) times and 2.11(4.22) times cheaper.

6 Conclusion

Fine-tuned PLMs have been shown to be vulnerable to OOD and adversarial inputs. The DM approach has been shown to improve model robustness (Swayamdipta et al., 2020); however, it is computationally expensive. In this paper, we have presented FTFT, a novel approach for fine-tuning PLMs which yields both better efficiency and better robustness over conventional ERM fine-tuning (§5). FTFT is built on the DM approach, based on two observations: 1) reference model training dynamics are highly transferable across different model sizes (§4.1) and pretraining methods (§4.2); and 2) models trained using DM learn faster than when using conventional ERM fine-tuning. We have also discussed the conditions for successful FTFT runs (§4.3). We believe that FTFT will be an important tool for future researchers and practitioners to perform efficient PLM fine-tuning, especially in situations where robustness against out-of-distribution or adversarial inputs is essential.

Limitations

Our work opens up several directions for future research. First, we have observed that effective reference models identify more instances as easy. This needs to be validated with more controlled experiments to identify or eliminate other possible model characteristics that play a role in the effectiveness of reference models. Second, we have only tested FTFT on two classification tasks. Future studies can extend our work to other types of tasks, such as generative tasks (e.g., question answering), to examine the generalizability of FTFT.

References

Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. 2017. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2020. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

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

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

Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. 2020. End-to-end bias mitigation by modelling biases in corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8706–8716, Online. Association for Computational Linguistics.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2022. WANLI: Worker and AI collaboration for natural language inference dataset creation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6826–6847, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. 2021. Just train twice: Improving group robustness without training group information. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6781–6792. PMLR.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. 2020. Learning from failure: Training debiased classifier from biased classifier. In *Advances in Neural Information Processing Systems*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial

NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems*.

Alan Ramponi and Sara Tonelli. 2022. Features or spurious artifacts? data-centric baselines for fair and robust hate speech detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3027–3040, Seattle, United States. Association for Computational Linguistics.

Victor Sanh, Thomas Wolf, Yonatan Belinkov, and Alexander M Rush. 2021. Learning from others' mistakes: Avoiding dataset biases without modeling them. In *International Conference on Learning Representations*.

Aviad Sar-Shalom and Roy Schwartz. 2023. Curating datasets for better performance with example training dynamics. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10597–10608, Toronto, Canada. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. 2019. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2020. Well-read students learn better: On the importance of pre-training compact models.

Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020. Mind the trade-off: Debiasing NLU models without degrading the in-distribution performance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8717–8729, Online. Association for Computational Linguistics.

Bertie Vidgen, Dong Nguyen, Helen Margetts, Patricia Rossini, and Rebekah Tromble. 2021a. Introducing CAD: the contextual abuse dataset. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2289–2303, Online. Association for Computational Linguistics.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021b. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1667–1682, Online. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin Lee, Bugra Akyildiz, Maximilian Balandat, Joe Spisak, Ravi Jain, Mike Rabbat, and Kim Hazelwood. 2022. Sustainable ai: Environmental implications, challenges and opportunities. In *Proceedings of Machine Learning and Systems*, volume 4, pages 795–813.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. 2023. Doremi: Optimizing data mixtures speeds up language model pretraining. In *NeurIPS 2023*.

Michael Zhang, Nimit S Sohoni, Hongyang R Zhang, Chelsea Finn, and Christopher Re. 2022. Correct-n-contrast: a contrastive approach for improving robustness to spurious correlations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 26484–26516. PMLR.

A Training Specifications

A.1 Experimental Setup

For training all models, we use AdamW (Loshchilov and Hutter, 2019) as the optimizer with a batch size of 32. We also use a linear learning rate scheduler with 10% warmup. For fine-tuning the small and base versions of both DeBERTa-V3 and ELECTRA, as well as TinyBERT, we use a learning rate of 2e-5. For DeBERTa-V3-Large and ELECTRA-Large, we respectively use 1e-5 and 2e-6 as the learning rates. We also experimented with 1e-5 and 5e-6 for ELECTRA-Large. However, we observed high ratios of failed runs, i.e., the training fails to converge and produces a worse than majority class baseline (Mosbach et al., 2021). We therefore follow the suggestions from Mosbach et al. (2021) to adopt a lower learning rate. We still encountered a few failed runs and we excluded them in our results. For the PoE baseline, we use a cross entropy loss weight of 0.3 and a PoE loss weight of 1.0; for the JTT baseline, we used SGD optimizer, `reduce_lr_on_plateau` scheduler, and respectively the third and the fifth epoch as the reference epoch for NLI and HSD. We up-weigh four times for each mispredicted sample.

We use Python 3.9 and PyTorch 2.0 for all experiments. For training PLMs, we use HuggingFace Transformers 4.32 (Wolf et al., 2020), Accelerate 0.22, and Datasets 2.14 (Lhoest et al., 2021). All experiments are performed on one NVIDIA A100 GPU. Training all models takes approximately seven GPU days.

A.2 Comparison of Training Costs

Here we show the training costs of different models (PFLOPs), of a single forward epoch. When taking back-propagation, optimization, and multiple epochs into consideration, these numbers should scale proportionally.

	NLI: MultiNLI	HSD: CAD
$\text{DeBERTaV3}_{\text{Small}}$	3116.93	312.35
$\text{DeBERTaV3}_{\text{Base}}$	6233.34	624.68
$\text{DeBERTaV3}_{\text{Large}}$	22160.84	2220.9
$\text{ELECTRA}_{\text{Small}}$	694.61	69.98
$\text{ELECTRA}_{\text{Base}}$	6227.14	627.37
$\text{ELECTRA}_{\text{Large}}$	22138.79	2230.5
TinyBERT	28.88	2.91

Table 4: Comparison of training costs, measured in PFLOPs. Here we only calculate the forward cost for a single epoch.

B Additional Results

Mode	Main Model	Ref. Model	CAD	DynaHate-Original			DynaHate-Perturb		
				R2	R3	R4	R2	R3	R4
ERM	DeBERTaV3$_{\text{Small}}$	-	$76.57_{0.74}$	$56.89_{5.13}$	$59.29_{3.89}$	$63.48_{0.99}$	$59.55_{2.67}$	$66.59_{1.51}$	$61.48_{1.30}$
ERM	DeBERTaV3$_{\text{Base}}$	-	$78.64_{0.55}$	$60.53_{2.22}$	$64.28_{0.83}$	$68.89_{2.00}$	$60.81_{1.56}$	$69.48_{1.30}$	$63.12_{2.02}$
ERM	DeBERTaV3$_{\text{Large}}$	-	$\mathbf{81.69_{0.57}}$	$75.44_{1.67}$	$73.32_{0.80}$	$76.12_{1.54}$	$70.62_{1.83}$	$77.41_{0.57}$	$68.89_{1.01}$
DM	DeBERTaV3$_{\text{Large}}$	Random	$76.22_{0.98}$	$63.38_{2.00}$	$61.59_{3.48}$	$71.21_{2.36}$	$64.05_{1.78}$	$72.10_{1.45}$	$62.88_{2.24}$
DM	DeBERTaV3$_{\text{Large}}$	DeBERTaV3$_{\text{Large}}$	$81.58_{0.72}$	$79.18_{1.12}$	$76.87_{1.89}$	$77.73_{1.42}$	$73.34_{1.14}$	$76.63_{0.94}$	$67.54_{0.50}$
Across different model sizes									
DM	DeBERTaV3$_{\text{Large}}$	DeBERTaV3$_{\text{Small}}$	$81.15_{0.33}$	$80.68_{3.14}$	$\mathbf{79.56_{0.64}}$	$\mathbf{79.86_{1.60}}$	$76.47_{0.98}$	$78.03_{0.16}$	$70.32_{0.94}$
DM	DeBERTaV3$_{\text{Large}}$	DeBERTaV3$_{\text{Base}}$	$80.12_{0.76}$	$80.12_{1.21}$	$76.34_{3.41}$	$78.82_{0.78}$	$74.60_{1.30}$	$77.81_{1.13}$	$68.67_{0.54}$
Across different pretraining methods									
DM	DeBERTaV3$_{\text{Large}}$	ELECTRA$_{\text{Small}}$	$79.74_{0.39}$	$78.09_{2.21}$	$77.40_{2.53}$	$78.75_{0.35}$	$75.26_{1.70}$	$76.79_{0.95}$	$70.05_{0.88}$
DM	DeBERTaV3$_{\text{Large}}$	ELECTRA$_{\text{Base}}$	$80.37_{0.43}$	$\mathbf{81.47_{1.62}}$	$78.17_{2.52}$	$78.38_{1.59}$	$\mathbf{76.99_{1.60}}$	$\mathbf{78.51_{1.08}}$	$\mathbf{71.01_{1.44}}$
DM	DeBERTaV3$_{\text{Large}}$	ELECTRA$_{\text{Large}}$	$79.48_{1.06}$	$76.71_{4.48}$	$75.97_{3.66}$	$78.58_{2.69}$	$73.55_{3.28}$	$77.80_{1.33}$	$69.81_{1.83}$
DM	DeBERTaV3$_{\text{Large}}$	BERT$_{\text{Large}}$	$79.95_{1.50}$	$79.10_{3.72}$	$75.87_{1.53}$	$77.64_{0.93}$	$72.94_{1.40}$	$77.09_{0.75}$	$67.47_{1.11}$
DM	DeBERTaV3$_{\text{Large}}$	RoBERTa$_{\text{Large}}$	$80.56_{0.60}$	$80.42_{1.26}$	$77.26_{1.87}$	$79.48_{1.58}$	$73.11_{0.68}$	$77.31_{1.20}$	$69.48_{1.06}$

Table 5: Transferability across different model sizes and pretraining methods, using DeBERTaV3 as the main model on HSD. We compare the performance (accuracy) of 1) DeBERTaV3 of different sizes fine-tuned using ERM, 2) DeBERTaV3$_{\text{Large}}$ as the main model, using a random DM (random 33% training instances), and DeBERTaV3$_{\text{Large}}$ as the reference model (Ref. Model) to construct a DM (original DM), 3) DeBERTaV3$_{\text{Large}}$ as the main model, using DeBERTaV3$_{\text{Small}}$ and DeBERTaV3$_{\text{Base}}$ as reference models to construct a DM, 4) DeBERTaV3$_{\text{Large}}$ as the main model, using ELECTRA of different sizes, BERT$_{\text{Large}}$ and RoBERTa$_{\text{Large}}$, as reference models to construct a DM. R2–R4 of DynaHate refer to different rounds of collected data. Training dynamics are transferable across different sizes and pretraining methods: Creating DMs using different reference model sizes and pretraining methods results in comparable performance.

Mode	Main Model	Ref. Model	CAD	DynaHate-Original			DynaHate-Perturb		
				R2	R3	R4	R2	R3	R4
ERM	TinyBERT	-	$71.72_{0.41}$	$43.19_{1.91}$	$49.60_{0.37}$	$52.88_{2.61}$	$43.21_{2.53}$	$57.50_{1.18}$	$52.91_{2.48}$
ERM	ELECTRA$_{\text{Small}}$	-	$74.65_{0.49}$	$48.93_{2.36}$	$56.93_{1.32}$	$58.93_{1.18}$	$55.75_{1.40}$	$61.84_{1.90}$	$59.52_{2.12}$
ERM	ELECTRA$_{\text{Base}}$	-	$76.43_{0.72}$	$62.13_{1.20}$	$60.71_{2.81}$	$61.72_{1.41}$	$55.42_{1.86}$	$64.64_{0.81}$	$62.05_{0.66}$
ERM	ELECTRA$_{\text{Large}}$	-	$75.81_{3.72}$	$70.07_{10.64}$	$62.56_{7.90}$	$70.27_{6.12}$	$63.91_{9.13}$	$70.45_{4.66}$	$67.02_{4.76}$
DM	DeBERTaV3$_{\text{Large}}$	TinyBERT	$78.91_{0.92}$	$71.68_{1.81}$	$71.69_{0.91}$	$76.52_{2.04}$	$71.22_{2.62}$	$75.92_{1.83}$	$68.23_{1.83}$
DM	DeBERTaV3$_{\text{Large}}$	ELECTRA$_{\text{Small}}$	$79.74_{0.39}$	$78.09_{2.21}$	$77.40_{2.53}$	$\mathbf{78.75_{0.35}}$	$75.26_{1.70}$	$76.79_{0.95}$	$70.05_{0.88}$
DM	DeBERTaV3$_{\text{Large}}$	ELECTRA$_{\text{Base}}$	$\mathbf{80.37_{0.43}}$	$\mathbf{81.47_{1.62}}$	$\mathbf{78.17_{2.52}}$	$78.38_{1.59}$	$\mathbf{76.99_{1.60}}$	$\mathbf{78.51_{1.08}}$	$\mathbf{71.01_{1.44}}$
DM	DeBERTaV3$_{\text{Large}}$	ELECTRA$_{\text{Large}}$	$79.48_{1.06}$	$76.71_{4.48}$	$75.97_{3.66}$	$78.58_{2.69}$	$73.55_{3.28}$	$77.80_{1.33}$	$69.81_{1.83}$
DM	ELECTRA$_{\text{Large}}$	ELECTRA$_{\text{Small}}$	$76.50_{1.65}$	$75.63_{1.86}$	$67.92_{2.10}$	$74.06_{1.12}$	$68.43_{0.96}$	$71.62_{1.02}$	$67.30_{1.11}$
DM	ELECTRA$_{\text{Large}}$	ELECTRA$_{\text{Base}}$	$75.88_{1.70}$	$75.52_{0.18}$	$65.00_{2.67}$	$72.89_{0.40}$	$67.94_{3.47}$	$70.34_{0.54}$	$67.94_{0.18}$
DM	ELECTRA$_{\text{Large}}$	ELECTRA$_{\text{Large}}$	$74.73_{2.23}$	$65.25_{7.89}$	$62.78_{5.51}$	$72.45_{1.02}$	$62.92_{6.46}$	$71.17_{1.85}$	$68.23_{2.02}$
DM	ELECTRA$_{\text{Large}}$	Random	$72.83_{1.70}$	$62.14_{1.51}$	$58.13_{4.41}$	$68.23_{2.82}$	$63.30_{2.84}$	$67.93_{1.02}$	$65.22_{2.13}$

Table 6: Performance of different models on HSD with conventional fine-tuning (ERM) and DM using the 33% most ambiguous data identified with different reference models. Random in Ref. Model means randomly selecting 33% of the train data. The rows marked in gray are the results for DM training where the transfer was not successful, and the corresponding reference models. Successful transfer requires the reference model to be reasonably strong: reference models of clearly worse performance lead to degraded OOD performance for the main models. Note that although using TinyBERT and DeBERTaV3$_{\text{Large}}$ respectively as the reference and the main model yields better performance than ELECTRA, that is due to the better performance of DeBERTaV3$_{\text{Large}}$ than ELECTRA$_{\text{Large}}$: it is still worse than other methods using ELECTRA$_{\text{Large}}$ as the main model, especially on OOD test sets.

	NLI: MultiNLI			HSD: CAD		
	25%	33%	50%	25%	33%	50%
$\text{ELECTRA}_{\text{Small}}$	55.91%	47.77%	35.84%	68.72%	62.10%	46.07%
$\text{ELECTRA}_{\text{Base}}$	69.10%	62.88%	46.62%	73.36%	65.54%	47.56%
$\text{ELECTRA}_{\text{Large}}$	66.91%	59.08%	45.00%	67.80%	60.03%	45.01%
$\text{DeBERTaV3}_{\text{Small}}$	67.08%	61.31%	46.08%	72.05%	65.00%	47.94%
$\text{DeBERTaV3}_{\text{Base}}$	71.39%	64.02%	46.90%	73.20%	65.80%	48.04%
$\text{DeBERTaV3}_{\text{Large}}$	72.97%	64.84%	46.97%	74.06%	65.89%	47.72%
TinyBERT	51.29%	38.44%	20.01%	63.58%	54.45%	40.20%

Table 7: Differences between effective and ineffective reference models. We show the percentages of easy data identified by the data maps of different models; cells marked in gray are the ratios of easy data identified by ineffective reference models. The column names indicate different thresholds for $q\%$ in the DM method. The key difference between effective and ineffective reference models lies in their ability to identify easy cases: compared with other models, fewer data points are identified as easy by TinyBERT on both tasks, and by $\text{ELECTRA}_{\text{Small}}$ on NLI.

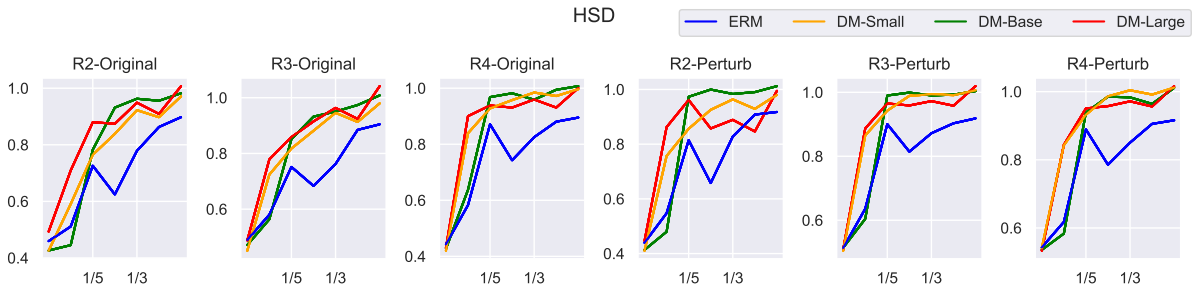

Figure 4: Performance on HSD when training the main model ($\text{ELECTRA}_{\text{Large}}$) with fewer training steps. ERM is standard ERM fine-tuning on the full training set. DM-* refers to fine-tuning $\text{ELECTRA}_{\text{Large}}$ with the DM method, using ELECTRA_{*} as the reference model. The X-axis is the percentile of training steps used, ranging from 1/15 to 7/15 of the total number of training steps. The Y-axis is the **percentage of performance compared with a model trained on the total number of training steps**. Fine-tuning the main model using data maps is much faster than ERM: the models achieve close-to-100% performance with only 1/3 of the training steps.

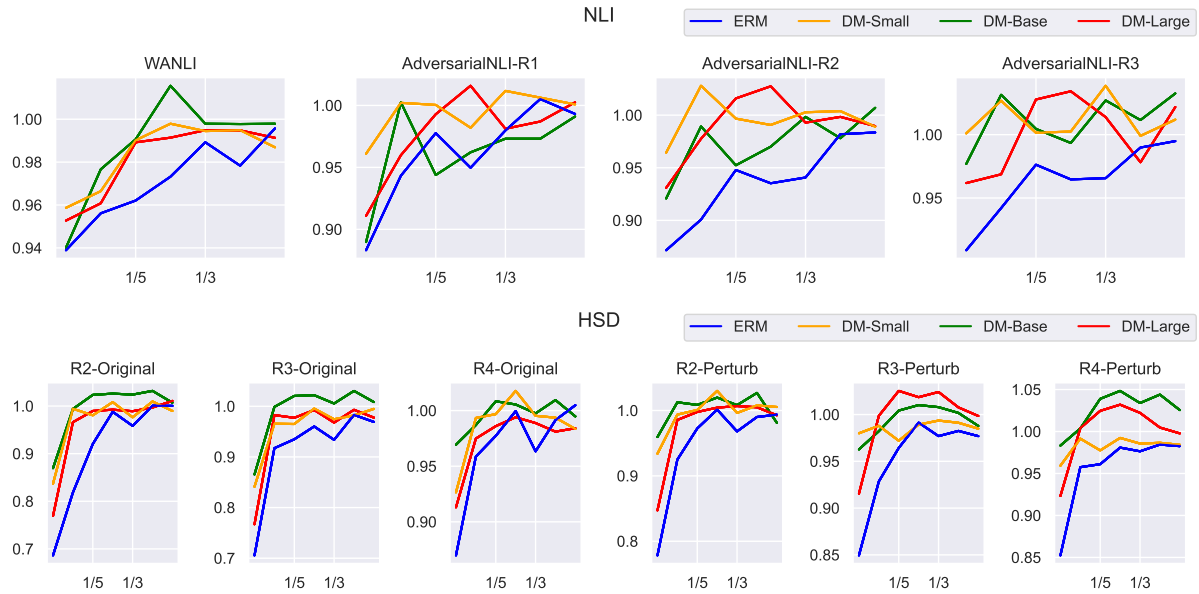

Figure 5: Performance on NLI (top) and HSD (bottom) when training the main model (DeBERTaV3) with fewer training steps. ERM is standard ERM fine-tuning on the full training set. DM-* refers to fine-tuning $\text{ELECTRA}_{\text{Large}}$ with the DM method, using reference model ELECTRA_{*}. The X-axis is the percentile of full training steps used, ranging from 1/15 to 7/15 of the total number of training steps. The Y-axis is the **percentile of performance compared with the full training steps**. Fine-tuning the main model using data maps is much faster than ERM: the models achieve close-to-100% performance using only 1/3 of the training steps.

Mode	Main Model	Ref. Model	Cost	MultiNLI	WANLI	AdversarialNLI R1	AdversarialNLI R2	AdversarialNLI R3
ERM	$\text{ELECTRA}_{\text{Large}}$	-	32.0	$\mathbf{90.75_{0.19}}$	$65.85_{0.41}$	$54.20_{1.13}$	$39.38_{1.17}$	$36.10_{0.38}$
DM	$\text{ELECTRA}_{\text{Large}}$	$\text{ELECTRA}_{\text{Large}}$	64.0	$90.33_{0.11}$	$65.37_{0.61}$	$53.73_{1.29}$	$39.67_{2.08}$	$36.17_{0.22}$
FTFT	$\text{ELECTRA}_{\text{Large}}$	$\text{ELECTRA}_{\text{Small}}$	11.7	$89.46_{1.24}$	$60.87_{0.57}$	$46.05_{1.37}$	$35.82_{1.11}$	$32.60_{0.82}$
FTFT	$\text{ELECTRA}_{\text{Large}}$	$\text{ELECTRA}_{\text{Base}}$	19.7	$90.06_{0.20}$	$\mathbf{65.85_{0.86}}$	$\mathbf{54.70_{1.31}}$	$\mathbf{40.23_{0.38}}$	$\mathbf{37.58_{1.17}}$

Mode	Main Model	Ref. Model	Cost	CAD	DynaHate-Original R2	DynaHate-Original R3	DynaHate-Original R4	DynaHate-Perturb R2	DynaHate-Perturb R3	DynaHate-Perturb R4
ERM	$\text{ELECTRA}_{\text{Large}}$	-	32.0	$75.81_{3.72}$	$70.07_{10.64}$	$62.56_{7.90}$	$70.27_{6.12}$	$63.91_{9.13}$	$70.45_{4.66}$	$67.02_{4.76}$
DM	$\text{ELECTRA}_{\text{Large}}$	$\text{ELECTRA}_{\text{Large}}$	64.0	$74.73_{2.23}$	$65.25_{7.89}$	$62.78_{5.51}$	$72.45_{1.02}$	$62.92_{6.46}$	$71.17_{1.85}$	$68.23_{2.02}$
FTFT	$\text{ELECTRA}_{\text{Large}}$	$\text{ELECTRA}_{\text{Small}}$	15.9	$75.52_{1.54}$	$73.45_{3.34}$	$\mathbf{66.54_{3.37}}$	$\mathbf{73.79_{2.06}}$	$67.08_{2.92}$	$\mathbf{72.05_{1.74}}$	$68.11_{0.78}$
FTFT	$\text{ELECTRA}_{\text{Large}}$	$\text{ELECTRA}_{\text{Base}}$	23.9	$\mathbf{77.15_{0.96}}$	$\mathbf{74.17_{2.33}}$	$65.53_{0.23}$	$73.43_{0.66}$	$\mathbf{68.77_{2.00}}$	$70.60_{0.92}$	$\mathbf{68.70_{1.97}}$

Table 8: Comparison between FTFT and ERM/original DM fine-tuning. Performance of ELECTRA on NLI (top, accuracy) and HSD (bottom, macro-F1). ERM is conventional ERM fine-tuning, and FTFT refers to using the 33% most ambiguous data identified by different reference models (i.e., Ref. Model). Cost refers to the fine-tuning cost, with the cost of fine-tuning ELECTRA-Small with ERM as the unit. FTFT yields both better efficiency and better robustness compared to both ERM fine-tuning and the original DM method. For NLI, we only train the main model for 1/3 of the total steps. For HSD, we observe that ELECTRA converges a bit slower, see Figure 4. We therefore use 7/15 of the total steps for FTFT to obtain better performance (still less than 1/2). FTFT yields both better efficiency and better robustness, compared to both ERM fine-tuning and the original DM method, except for using $\text{ELECTRA}_{\text{Small}}$ as the reference model on NLI.