FTFT: EFFICIENT AND ROBUST FINE-TUNING BY TRANSFERRING TRAINING DYNAMICS

Yupei Du, Albert Gatt & Dong Nguyen
Utrecht University
Utrecht, the Netherlands
{y.du,a.gatt,d.p.nguyen}@uu.nl

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 improves 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 PLMs. 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 (ERM). 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 PLMs. However, fine-tuning large PLMs is also more expensive, 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.

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

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

First, we study the following unexplored question: Are data maps transferable across different model sizes and 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*. The motivation is two-fold: first, smaller reference models are more computationally efficient; second, the more capable a reference model is, the better it is at memorizing training data quickly (Tirumala et al., 2022; Carlini et al., 2023), making such models less suitable to identify ambiguous or hard training instances. We make three key observations. First, in most cases, *training dynamics are highly transferable* across different model sizes (§4.1) and pretraining methods (§4.2). 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 (§4.3). 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 the traditional approach based on Empirical Risk Minimization (ERM), i.e., good performance is achieved using fewer steps (§5).

Second, building on these findings, we propose **Fine-Tuning by transFerring Training dynamics (FTFT, §5)**: an efficient fine-tuning approach that leads to improved OOD 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 train the main model. Experiments on two tasks, NLI and HSD, show that FTFT achieves better performance on OOD input than ERM, while lowering the training cost by > 2 times.

2 BACKGROUND

Dual-Model Approaches for Robustness Many studies have proposed dual-model approaches to improve model robustness, without the knowledge of instance group information. Nam et al. (2020) first train a reference model using generalized cross-entropy loss, and then train a main model while assigning higher weights to 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 weak 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 use training dynamics instead of correctness to categorize training instances. 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 & 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 investi-

gate either DM transfer across model sizes and pretraining methods, or how such transferability can be exploited to improve 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 and improve ID performance. 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). Following Swayamdipta et al. (2020), we set $q\% = 33\%$.

Data To study model robustness, we include a few challenging OOD test sets for each task, in addition to the training set and an ID validation set.

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. Following Ramponi & Tonelli (2022), we frame the task as a binary classification task, by marking identity-related abuse as hateful and other categories as non-hateful. As OOD test sets, we use DynaHate (Vidgen et al., 2021b), which contains three rounds of adversarial data collection and perturbations, because it is challenging and aligns with CAD's hate speech definition.

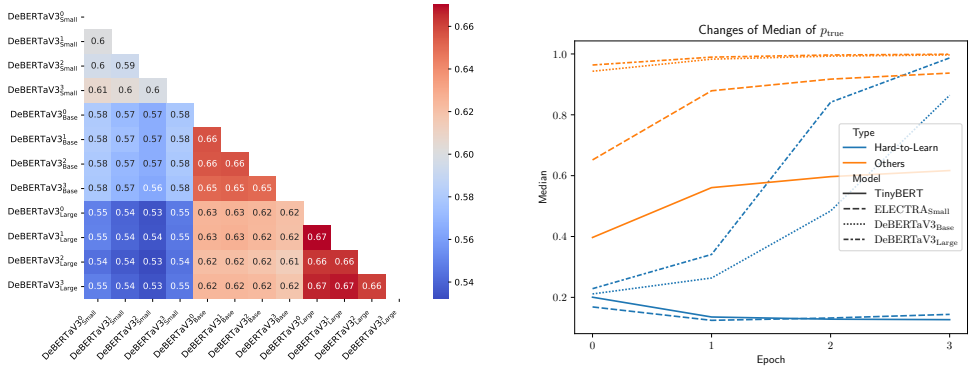
Models We mainly use DeBERTaV3 (He et al., 2023) and ELECTRA (Clark et al., 2020) in our experiments, because they offer strong performance and various model sizes. To study the transferability across different pretraining methods, we also use TinyBERT (Turc et al., 2020), BERT (Devlin et al., 2019)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

Training We train all models for 60k (NLI) steps and 6k (HSD) steps. To inspect the training dynamics, we checkpoint the model 15 times during training, i.e., every 4k steps for NLI and every 400 steps for HSD. For other hyper-parameters, we use the recommended values from the PLM papers without further tuning. We use four different random seeds for each run. Full training details (e.g., optimization, software and hardware) and discussions 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 $\text{DeBERTaV3}_{\text{Large}}$).

[2]Costs for fine-tuning different PLMs are in Appendix A.1. We report PFLOPs rather than GPU hours because we noticed occasional low GPU utilization especially when fine-tuning smaller PLMs.



(a) Consistency across different sizes

(b) Change of median p_{true}

Figure 1: Figure 1a: Consistency across different sizes of DeBERTaV3 on NLI. The numbers are the percentages (0–1) of the ambiguous training instances shared by two models. Training dynamics are transferable across different sizes: the percentages of shared ambiguous instances between models of different sizes are only slightly smaller than those between models of different random seeds (shown as superscript). Figure 1b: Change of median p_{true} over different training epochs on MultiNLI.

We focus on these issues for two reasons. First, transferability across model sizes enables using more efficient (and usually less capable) reference models, which 1) can improve training efficiency, and 2) makes it easier to identify ambiguous/hard instances, because less capable models are usually worse at memorizing training instances. Second, transferability across pretraining methods can help achieve the advantages of using efficient reference models even in cases where more efficient variants of the main pretraining method are unsuitable or unavailable for the task. Moreover, understanding transferability can shed 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.

We define successful transfers as transfers that produce comparable or better OOD performance than training the main model with ERM. Our results show that training dynamics are transferable [3] with a few exceptions. To understand the conditions for successful transfers, we analyze the failure cases (§4.3), and find that the DMs of reference models that lead to successful transfers typically contain more training instances in the easy subset. This finding can serve as a guideline for choosing reference models without training the main model, which is 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 (e.g. DeBERTaV3$_{\text{Small}}$ as the reference model for DeBERTaV3$_{\text{Large}}$). Successful transfers of this type enables the use of more efficient reference models.

The results for DeBERTaV3 are shown in Table 1 (NLI) and Table 6 (HSD, Appendix C). We make three observations. First, using larger main models results in better performance: almost all methods using DeBERTaV3$_{\text{Large}}$ as the main model outperform those using DeBERTaV3$_{\text{Small}}$ and DeBERTaV3$_{\text{Base}}$. Second, consistent with Swayamdipta et al. (2020), ERM achieves the best ID performance, while DM performs the best on OOD data. Third and most importantly, *training dynamics are transferable across different model sizes*: when using DeBERTaV3$_{\text{Large}}$ as the main model, changing the reference model to DeBERTaV3$_{\text{Small}}$ or DeBERTaV3$_{\text{Base}}$ yields comparable or even better performance. This observation is consistent with our hypothesis that efficient models are more sensitive to ambiguous and difficult instances. As a result, they can serve as more effective reference models compared to larger, more capable models.

To investigate transferability of DMs further, we analyze whether reference models of different sizes identify similar groups of ambiguous instances. Figure 1a shows the percentage of ambiguous

[3] To further validate our observations, we include a significance test in Appendix B.

Mode	Main Model	Ref. Model	MultiNLI	WANLI	AdversarialNLI		
					R1	R2	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) different sizes of DeBERTaV3 fine-tuned using ERM, and DeBERTaV3$_{\text{Large}}$ as the main model, using 2) random 33% training instances, 3) DeBERTaV3$_{\text{Large}}$ as reference model (Ref. Model), 4) DeBERTaV3$_{\text{Small}}$ and DeBERTaV3$_{\text{Base}}$ as reference models, and 5) different sizes of ELECTRA, BERT$_{\text{Large}}$ and RoBERTa$_{\text{Large}}$, as reference models. R1–R3 in AdversarialNLI refer to different rounds of collected data. Training dynamics are transferrable across different sizes and pretraining methods: constructing DMs using different reference models results in comparable performance.

instances shared by reference models of different sizes or random seeds: the percentages 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.[4]

4.2 Transferability Across Pretraining Methods

We now study the transferability of training dynamics across different pretraining methods. If these transfers are successful, we can still exploit efficient DM-based transfer in case there is no smaller or more efficient version of the main model that suits the downstream task.

The results for DeBERTaV3$_{\text{Large}}$ as the main model with different reference models are shown in Table 1 (NLI) and Table 6 (HSD, Appendix C). *Across pre-training methods, training dynamics are generally transferable*: DeBERTaV3$_{\text{Large}}$ achieves comparable performance using DMs constructed by different reference models in most cases. However, there are a few exceptions: when using ELECTRA$_{\text{Small}}$ as the reference model, the performance is clearly worse on the 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 zooming in 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, four models fine-tuned with ERM: the small, base, and large versions of ELECTRA, and TinyBERT. We use these models because they have clearly different sizes and capabilities (ELECTRA$_{\text{Large}}$ >

[4]The expected shared percentage of random DMs is 0.33, as we select the 33% most ambiguous instances.

Mode	Main Model	Ref. Model	MultiNLI	WANLI	AdversarialNLI		
					R1	R2	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 ERM and DM using the 33% most ambiguous data points identified with different reference models. The gray-shaded rows are the 1) unsuccessful DM transfers and 2) the corresponding reference models used in these transfers. 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.

$\text{ELECTRA}_{\text{Base}} > \text{ELECTRA}_{\text{Small}} > \text{TinyBERT}$). Second, we use these models as reference models for $\text{DeBERTaV3}_{\text{Large}}$. 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 $\text{ELECTRA}_{\text{Large}}$ 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. Random DM is included as a baseline. The results are shown in Table 2 (NLI) and and Table 7 (HSD, Appendix C). We make two observations.

First, reference models with very poor ID performance (e.g., TinyBERT and $\text{ELECTRA}_{\text{Small}}$) lead to failed transfers. Generally, for these ineffective reference models, poorer reference models lead to worse main model OOD performance: reference model TinyBERT show worse main model performance than $\text{ELECTRA}_{\text{Small}}$. Moreover, the success of transfers mostly depends on the reference model rather than the main model: transfers from $\text{ELECTRA}_{\text{Small}}$ to both models are unsuccessful.

Second, weak reference models do not negatively affect ID performance much. For instance, transfers from $\text{ELECTRA}_{\text{Small}}$ to $\text{DeBERTaV3}_{\text{Large}}$ yield the best accuracy on MultiNLI. We suspect the reason is that weak models usually identify simple training instances as ambiguous data, which are found 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 that instances in our training set exhibit varying levels of difficulty (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 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 these instances have both high means and low standard deviations for p_{true}. Meanwhile, 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, this larger overlap means more instances will be identified as easy.

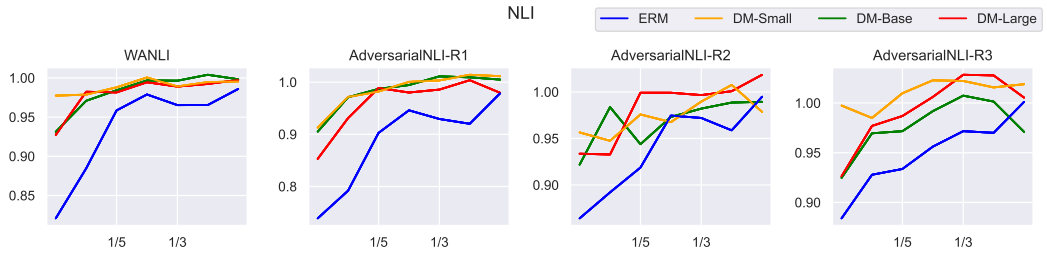


Figure 2: Performance on NLI when training the main model (ELECTRA$_{\text{Large}}$) with fewer training steps. ERM is standard ERM fine-tuning on the full training set. DM-* refers to fine-tuning ELECTRA$_{\text{Large}}$ with the DM approach, using ELECTRA$_{*}$ as the reference model. The X-axis is 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.

We now validate our reasoning. Given a reference model, we first split the training instances into two subsets based on mean p_{true}: hard-to-learn (10% of training instances) and others (the remaining 90%). These two subsets represent difficult and simple 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 1b shows our results on MultiNLI, using two effective (DeBERTaV3$_{\text{Large}}$ and DeBERTaV3$_{\text{Base}}$) and two ineffective (ELECTRA$_{\text{Small}}$ and TinyBERT) reference models. With effective reference models, 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 ineffective reference models, 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.

To further validate our reasoning, we compute the percentages of training instances identified as easy by different reference models (Table 8 in Appendix C): ineffective reference models indeed identify fewer data points as easy. 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% as easy training instances (the maximum is 50%, when ambiguous and hard-to-learn data align perfectly).

5 Towards Efficient and Robust Fine-Tuning

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 still need to fine-tune the main model using ERM, the DM approach remains less efficient than just fine-tuning a model with only ERM. In this section, we address this limitation by showing that with DM we need fewer steps to fine-tune the main model. We therefore propose a novel approach, **F**ine-**T**uning by trans**F**erring **T**raining dynamics (FTFT), which consistently offers better efficiency and robustness over ERM.

DM Trains Faster Figure 2 shows 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, compared with the ERM baseline. The HSD results are in Figure 3 (Appendix C). We show the relative performance against that achieved with all training steps. We observe that using DM leads to much faster learning than ERM on all OOD datasets. With DM, only 1/3 of the total number of training steps already achieves almost 100% performance. 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 C and observe similar trends.

FTFT: Achieving both Efficiency and Robustness FTFT involves two crucial changes to the DM approach, 1) more efficient PLMs as reference models, and 2) 1/3 of the training steps used

Mode	Main Model	Ref. Model	Cost	MultiNLI	WANLI	AdversarialNLI		
						R1	R2	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	CAD	DynaHate-Original			DynaHate-Perturb		
				R2	R3	R4	R2	R3	R4
ERM	$\text{DeBERTaV3}_{\text{Large}}$	-	$81.69_{0.57}$	$75.44_{1.67}$	$73.32_{0.80}$	$76.12_{1.54}$	$70.62_{1.83}$	$77.41_{0.57}$	$68.89_{1.01}$
DM	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Large}}$	$81.58_{0.72}$	$79.18_{1.12}$	$76.87_{1.89}$	$77.73_{1.42}$	$73.34_{1.14}$	$76.63_{0.94}$	$67.54_{0.50}$
JTT	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Large}}$	$81.17_{0.69}$	$75.54_{2.88}$	$72.80_{3.16}$	$76.70_{1.57}$	$72.45_{1.97}$	$76.11_{1.09}$	$67.92_{1.26}$
PoE	$\text{DeBERTaV3}_{\text{Large}}$	TinyBERT	$\mathbf{81.70_{0.76}}$	$76.87_{1.19}$	$73.48_{1.29}$	$76.58_{1.25}$	$70.46_{1.37}$	$77.17_{0.61}$	$68.28_{1.09}$
FTFT	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Small}}$	$80.73_{0.77}$	$78.77_{2.39}$	$\mathbf{77.53_{3.11}}$	$\mathbf{79.48_{1.52}}$	$\mathbf{76.19_{1.93}}$	$77.53_{0.82}$	$69.31_{1.69}$
FTFT	$\text{DeBERTaV3}_{\text{Large}}$	$\text{DeBERTaV3}_{\text{Base}}$	$79.76_{1.31}$	$\mathbf{82.05_{2.46}}$	$76.77_{2.19}$	$78.62_{1.23}$	$75.22_{2.64}$	$\mathbf{78.43_{1.86}}$	$\mathbf{71.00_{3.15}}$

Table 3: Performance of DeBERTaV3 on NLI (top, accuracy) and HSD (bottom, macro-F1) using different methods. FTFT refers to using the 33% most ambiguous data instances *and only training for 1/3 of the total steps*. Cost includes both the reference model and the main model, 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. The relative cost for different approaches for HSD is the same as the ones for NLI.

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 training 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 $\text{DeBERTaV3}_{\text{Large}}$ as the main model, and $\text{DeBERTaV3}_{\text{Small}}$ and $\text{DeBERTaV3}_{\text{Base}}$ as reference models. We compare FTFT against ERM, 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 $\text{ELECTRA}_{\text{Small}}$ with ERM as unit (the relative cost for HSD is the same as NLI). Note that our cost calculation considers 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 OOD datasets. Second, FTFT enjoys higher efficiency than other methods. For example, FTFT with $\text{DeBERTaV3}_{\text{Base}}$ and $\text{DeBERTaV3}_{\text{Small}}$ as reference models only cost 19.7 and 15.2 units of computation: they are respectively 1.63 and 2.11 times cheaper than ERM, and 3.26 and 4.22 times cheaper than DM.

6 Conclusion and Limitations

Fine-tuned PLMs have been shown to be vulnerable to OOD and adversarial input. The DM approach can 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 ERM (§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 ERM. 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 is essential.

Our work opens up several directions for future research. First, we have observed that effective reference models identify more instances as easy. This needs further empirical validation with more controlled experiments, as well as theoretical understandings of DMs, to identify or eliminate other potential factors. Second, we have only developed FTFT for classification tasks. Future studies can extend our work to other tasks, such as generation (e.g., question answering) and unsupervised tasks (e.g., language modeling), to examine the generalizability of FTFT.

REPRODUCIBILITY STATEMENT

We will release our code and data upon publication. Our code is based on open-source libraries and open datasets, and we will provide instructions for reproducing our results.

REFERENCES

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TatRHT_1cK.

Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/2f37d10131f2a483a8dd005b3d14b0d9-Paper.pdf.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020. URL https://openreview.net/pdf?id=r1xMH1BtvB.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJg2b0VYDr.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sE7-XhLxHA.

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

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

Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. End-to-end bias mitigation by modelling biases in corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8706–8716, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.769. URL https://aclanthology.org/2020.acl-main.769.

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. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical*

Methods in Natural Language Processing: System Demonstrations, pp. 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.emnlp-demo.21.

Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. WANLI: Worker and AI collaboration for natural language inference dataset creation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 6826–6847, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-emnlp.508.

Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6781–6792. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/liu21f.html.

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

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

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=nzpLWnVAyah.

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

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

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Uj7pF-D-YvT.

Alan Ramponi and Sara Tonelli. 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*, pp. 3027–3040, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.221. URL https://aclanthology.org/2022.naacl-main.221.

Victor Sanh, Thomas Wolf, Yonatan Belinkov, and Alexander M Rush. Learning from others' mistakes: Avoiding dataset biases without modeling them. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Hf3qXoiNkR.

Aviad Sar-Shalom and Roy Schwartz. Curating datasets for better performance with example training dynamics. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10597–10608, Toronto, Canada, July 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.findings-acl.674.

Thibault Sellam, Steve Yadlowsky, Ian Tenney, Jason Wei, Naomi Saphra, Alexander D'Amour, Tal Linzen, Jasmijn Bastings, Iulia Raluca Turc, Jacob Eisenstein, Dipanjan Das, and Ellie Pavlick.

The multiBERTs: BERT reproductions for robustness analysis. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=K0E_F0gFDgA.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1355. URL https://aclanthology.org/P19-1355.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9275–9293, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.746. URL https://aclanthology.org/2020.emnlp-main.746.

Damien Teney, LIN Yong, Seong Joon Oh, and Ehsan Abbasnejad. ID and OOD performance are sometimes inversely correlated on real-world datasets. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=HZQZli6amV.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=u3vEuRr08MT.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models, 2020. URL https://openreview.net/forum?id=BJg7x1HFvB.

Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 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*, pp. 8717–8729, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.770. URL https://aclanthology.org/2020.acl-main.770.

Bertie Vidgen, Dong Nguyen, Helen Margetts, Patricia Rossini, and Rebekah Tromble. 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*, pp. 2289–2303, Online, June 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.182. URL https://aclanthology.org/2021.naacl-main.182.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 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)*, pp. 1667–1682, Online, August 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.132. URL https://aclanthology.org/2021.acl-long.132.

Adina Williams, Nikita Nangia, and Samuel Bowman. 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)*, pp. 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

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. Sustainable ai: Environmental implications, challenges and opportunities. In D. Marculescu, Y. Chi, and C. Wu (eds.), *Proceedings of Machine Learning and Systems*, volume 4, pp. 795–813, 2022. URL https://proceedings.mlsys.org/paper_files/paper/2022/file/462211f67c7d858f663355eff93b745e-Paper.pdf.

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

Michael Zhang, Nimit S Sohoni, Hongyang R Zhang, Chelsea Finn, and Christopher Re. Correct-n-contrast: a contrastive approach for improving robustness to spurious correlations. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 26484–26516. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/zhang22z.html.

A Training Specifications

A.1 Experimental Setup

Optimization For training all models, we use AdamW (Loshchilov & 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, following Sanh et al. (2021), we use a cross entropy loss weight of 0.3 and a PoE loss weight of 1.0; for the JTT baseline, we used the SGD optimizer, `reduce_lr_on_plateau` scheduler, and respectively the third and the fifth epoch[5] as the reference epoch for NLI and HSD. We up-weigh four times for each mispredicted sample.

Number of Training Steps Following Swayamdipta et al. (2020), we train our models on MultiNLI for approximately five epochs (60,000 steps). Moreover, because we are not aware of previous studies that use DM on CAD, we follow the recommendation of He et al. (2023) and Clark et al. (2020) to train our models on CAD for approximately 10 epochs (6,000 steps).

However, in our follow-up experiments, we observe that these previously determined training durations may not be optimal. In §5 we show that for Empirical Risk Minimization (ERM), training for 20,000 and 2,000 steps led to inferior performance compared to training for 60,000 and 6,000 steps. However, this comparison overlooks the impact of the learning rate scheduler and warmup phase. In other words, halting the training at 20000 and 2000 steps differs from conducting the training exclusively for these durations. Specifically, when we limit the training to just 20,000 steps for NLI and 2,000 steps for HSD, the performance of ERM matches that of the longer ERM training periods of 60,000 and 6,000 steps. Nevertheless, our observation that DM learns faster than ERM training still holds in this experiment: 1) when we train our models with DM/FTFT approaches using different reference models for a shorter duration of 20,000 and 2,000 steps (limit the training durations instead of halting), it consistently outperforms those trained with ERM using the same training duration; 2) when setting the number of training steps to 20,000 steps and 2,000 steps, we observe DM/FTFT approaches achieve $\sim 100\%$ performance on NLI and HSD with less than 10,000 steps and 1,000 steps. We expect to investigate the optimal number of training steps for FTFT in future work.

Software and Hardware 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

We show the training costs for different models of a forward epoch. When taking back-propagation, optimization, and multiple epochs into consideration, these numbers should scale proportionally.

B Significance Test

We conduct significance tests on our results in Table 1 and Table 6. Because we use four random seeds, common statistical tests do not suit our needs here. We therefore use MultiBootstrap (Sellam

[5]For NLI, we follow the hyper-parameter choices from Liu et al. (2021). For HSD, we also mostly follow the NLI hyper-parameters except for reference epoch. We mostly follow previous studies because it is not clear which objective to use for tuning hyper-parameters for OOD performance, since ID validation dataset performance does not always correlate positively with OOD performance, and sometimes even show a trade-off (Teney et al., 2023). We use a slightly later reference epoch for HSD because CAD (HSD train data) is smaller than MultiNLI (NLI train data), and takes more epochs to train (but fewer number of steps in each epoch). Slightly later reference epoch therefore could be more appropriate.

	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	2220.90
$\text{ELECTRA}_{\text{Small}}$	694.61	69.98
$\text{ELECTRA}_{\text{Base}}$	6227.14	627.37
$\text{ELECTRA}_{\text{Large}}$	22138.79	2230.50
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.

Task	Ref. Model	Mean	p-value
HSD	DeBERTV3-Small	0.06	**0.0002**
HSD	DeBERTV3-Base	0.04	**0.0002**
HSD	DeBERTV3-Large	0.02	**0.005**
HSD	BERT-Large	0.02	**0.0168**
HSD	RoBERTa-Large	0.04	**0.0012**
HSD	ELECTRA-Small	0.04	**0.0002**
HSD	ELECTRA-Base	0.06	**0.0002**
HSD	ELECTRA-Large	0.03	**0.0322**

Task	Ref. Model	Mean	p-value
NLI	DeBERTV3-Small	0.48	*0.0578*
NLI	DeBERTV3-Base	0.65	**0.0120**
NLI	DeBERTV3-Large	0.20	0.2288
NLI	BERT-Large	1.09	**0.0016**
NLI	RoBERTa-Large	0.36	0.0916
NLI	ELECTRA-Base	0.48	*0.0552*
NLI	ELECTRA-Large	-0.03	0.4984

Table 5: MultiBootstrap results when comparing DM with $\text{DeBERTaV3}_{\text{Large}}$ as the main model versus ERM, on HSD and NLI. All HSD results and 4 out 7 methods in NLI are either significantly (p-value $<$ 0.05, in boldface) or marginally significantly (0.05 $<=$ p-value $<$ 0.06, in italics) better than ERM training.

et al., 2022), which is designed specifically for comparing models that have been trained with different random seeds, by generating bootstrap samples from both test instances and random seeds. To gain a comprehensive overview, we combine all out-of-distribution test data for each task. Specifically, we use 5000 samples for bootstrapping to compare each DM-based method against ERM. We observe (Table 5) that all HSD results and 4 out 7 methods in NLI are either significantly (p-value $<$ 0.05, in boldface) or marginally significantly (0.05 $<=$ p-value $<$ 0.06, in italics) better than ERM training.

C Additional Results

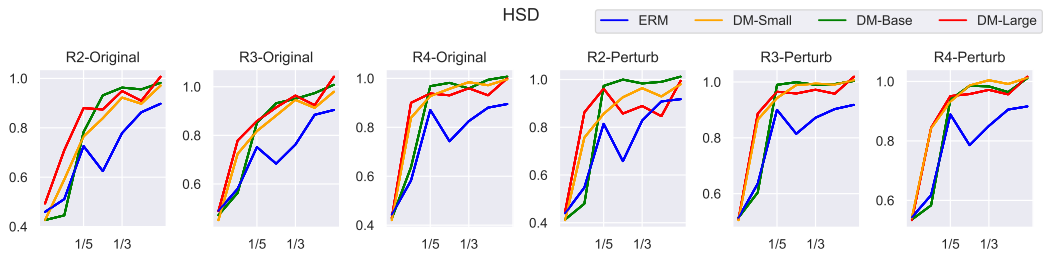


Figure 3: Performance on HSD when training the main model ($\text{ELECTRA}_{\text{Large}}$) with fewer training steps. ERM is standard ERM fine-tuning on the full training set. DM-* refers to fine-tuning $\text{ELECTRA}_{\text{Large}}$ with the DM method, using ELECTRA_{*} as the reference model. The 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.

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 6: Transferability across different model sizes and pretraining methods, using DeBERTaV3 as the main model on HSD. We compare the performance (accuracy) of 1) different sizes of DeBERTaV3 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 different sizes of ELECTRA, 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 7: Experiments on HSD to understand the conditions for successful transfer. The table shows the 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 8: Differences between effective and ineffective reference models. We show the percentages data identified as easy 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 the ratio of instances that they identify as easy: compared with other models, fewer data points are identified as easy by TinyBERT on both tasks, and by $\text{ELECTRA}_{\text{Small}}$ on NLI.

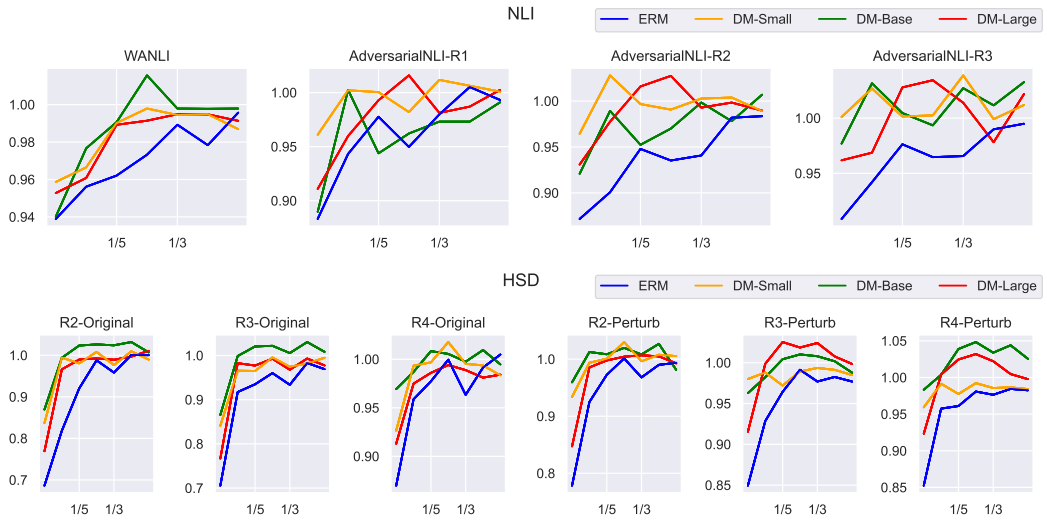


Figure 4: Performance on NLI (top) and HSD (bottom) when training the main model (DeBERTaV3) with fewer training steps. ERM is standard ERM fine-tuning on the full training set. DM-* refers to fine-tuning $\text{ELECTRA}_{\text{Large}}$ with the DM method, using reference model ELECTRA_{*}. The X-axis 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	R2	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			DynaHate-Perturb		
					R2	R3	R4	R2	R3	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 9: 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 3. 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.