Curating Datasets for Better Performance with Example Training Dynamics

Aviad Sar-Shalom[♡♠] **Roy Schwartz**[♠]
[♡] School of Computer Science & Engineering, Hebrew University of Jerusalem
[♠] Blavatnic School of Computer Science, Tel-Aviv University
{aviad.sar-shalom,roy.schwartz1}@mail.huji.ac.il

Abstract

The landscape of NLP research is dominated by large-scale models training on colossal datasets, relying on data quantity rather than quality. As an alternative to this landscape, we propose a method for weighing the relative importance of examples in a dataset based on their *Example Training dynamics* (ETD; Swayamdipta et al., 2020), a set of metrics computed during training. We propose a new way of computing the ETD of a dataset, and show that they can be used to improve performance in both in-distribution and out-of-distribution testing. We show that ETD can be transferable, i.e., they can be computed once and used for training different models, effectively reducing their computation cost. Finally, we suggest an active learning approach for computing ETD during training rather than as a preprocessing step—an approach that is not as effective, but dramatically reduces the extra computational costs.

1 Introduction

Breakthroughs in NLP are often the result of scaling up existing models in size and depth, and perhaps even more importantly—data (Hoffmann et al., 2022). To improve data quality, it has become common practice to train models on data that has been cleaned to some extent using simple heuristics (e.g. by removing non-language tokens), but not otherwise optimized for better performance. While some data-filtering approaches have been suggested to improve Out-Of-Distribution (OOD) generalization (Le Bras et al., 2020, Swayamdipta et al., 2020), they usually result in a decrease of In-Distribution (ID) performance.

We propose a method for curating datasets for better performance in both ID and OOD testing, enhancing data quality rather quantity. Our method is orthogonal to model architecture or size, and as such can be used alongside any LM to further improve results over specific tasks.

To implement our method we use the concept of Example Training Dynamics (ETD; Swayamdipta et al., 2020), which builds on the idea that the training process of models sheds light on the relative importance of specific examples within the datasets used. Specifically, Swayamdipta et al. (2020) have shown that over several epochs of training, a model may predict some examples in a dataset less consistently than others, and that those "ambiguous" examples are important for OOD generalization.

We propose a new method for computing ETD, as well as a new paradigm for using them in training. We show that by computing ETD over separate training processes (rather than over consecutive epochs of the same training process), and using ETD to weigh the importance of each example in the dataset, we can train a DeBERTa model (He et al., 2020) on the weighted versions of several NLI and multiple-choice datasets, improving average performance by 0.35% for ID testing and by 0.95% for OOD.

We next demonstrate that ETD can be transferable across models, i.e., ETD computed from the training process of model M_1 can be used to weigh a dataset, and train model M_2 on it with improved results, where M_2 differs from M_1 in initial weights, structuring details and pre-training scheme. Though we only show that for a specific use-case, if transferability of ETD holds generally, it may allow us to create weighted versions of datasets once, and use them for multiple training scenarios, including that of future models.

Finally, we propose Dynamic Training, a method for training a model while computing ETD and reweighing the training set between epochs. This method performs on par with our transferability method while requiring no additional compute beyond that of standard training, which makes it even more applicable under low computational budgets.

Our proposed method of computing ETD may allow practitioners to get more value out of their

Findings of the Association for Computational Linguistics: ACL 2023, pages 10597–10608

existing datasets, and pave the way towards similar methods to be used for improving large scale language model training. We publicly release our code, as well the weighted versions of the datasets used in this work.[1]

2 Example Training Dynamics

2.1 Background: Dataset Cartography

Our method aims to expand and improve on Dataset Cartography (Swayamdipta et al., 2020), a method for visualizing and characterizing the different training examples in a given dataset. Dataset Cartography uses Example Training Dynamics (ETD), which are metrics derived by examining the probability distribution that a model assigns to the possible labels for each example in a dataset, and following the changes in that distribution over several epochs of training. ETD derives two metrics for each example in the dataset: **Confidence**, which is the mean score the model assigns to the true label across epochs, and **Variability**, which is the standard deviation of that score. Swayamdipta et al. (2020) have shown that training a new model only on examples with high variability improves performance in OOD testing, as well as shortens training time. However, this comes at a slight cost in ID testing performance.

In this work, we propose a new method for computing ETD, which differs from that of Swayamdipta et al. (2020) in two ways: First, rather than following several epochs of a single training process to compute the metrics, we compute them by observing separate training processes of one epoch each. Second, we use the variability metric differently; rather than training a new model only on high-variability examples, we train it on the entire dataset, while emphasizing high-variability examples in training. We formally define our method and compare it to Swayamdipta et al. (2020) below.

2.2 Computation of Example Training Dynamics

We introduce a new method for working with Example Training Dynamics, which has two separate components: the computation of ETD, and their application in training new models. Fig. 1 illustrates the full pipeline of the two components as described below.

[1]Available at https://github.com/schwartz-lab-NLP/ETD

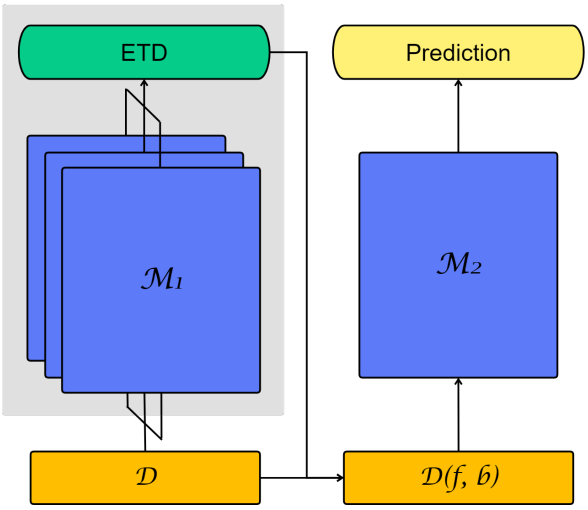

Figure 1: Computation and Usage of ETD. Multiple copies of model $\mathcal{M}_1$ are trained on a dataset $\mathcal{D}$, each for one epoch. The probability scores $\mathcal{M}_1$ outputs are used to compute ETD. The ETD are then used to to curate $\mathcal{D}_{(f,b)}$, a new version of $\mathcal{D}$, which reweighs the examples in $\mathcal{D}$. A (potentially different) model $\mathcal{M}_2$ trains on $\mathcal{D}_{(f,b)}$.

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ be a dataset of N examples x_i with corresponding labels y_i, and let $\mathcal{M}$ be a model with initial parameters θ that defines a probability distribution over the possible labels for examples in $\mathcal{D}$.

To compute the ETD of $\mathcal{D}$ with respect to $\mathcal{M}$, we train $\mathcal{M}$ on $\mathcal{D}$ for one epoch, and save the probabilities $\mathcal{M}$ assigns to the possible labels of each example in $\mathcal{D}$. We repeat the process E times with the same experimental setting at each iteration, except for the random seed, and for the ordering of the examples in $\mathcal{D}$, which is random and different at each iteration. After the first time $\mathcal{M}$ sees an example during training, it learns a *bias* towards its true label, and, if the same example is encountered again, this bias might affect the probability assigned to the different labels. To prevent this kind of bias and compute more informative ETD, the parameters θ are being reset between iterations.

Using the probabilities accumulated over all E iterations, we compute the two ETD metrics. The **confidence** of an example x_i w.r.t. $\mathcal{M}$ is defined as the mean probability $\mathcal{M}$ assigns to x_i's true label, y_i^*, across iterations:

$$\hat{\mu}_i = \frac{1}{E}\Sigma_{e=1}^{E} P_{\mathcal{M}_e}(y_i^*|x_i)$$

Where $P_{\mathcal{M}_e}(y_i^*|x_i)$ is the probability $\mathcal{M}$ assigns to the gold label y_i^* in iteration e. In a slight abuse of notation, we use the term $P_{\mathcal{M}_e}$, though in practice

the probability function P may change between examples even within the same iteration, as training progresses and the parameters of $\mathcal{M}$ change.[2]

The **variability** of an example w.r.t. $\mathcal{M}$ is defined as the standard deviation of said probability:

$$\hat{\sigma}_i = \sqrt{\frac{1}{E}\Sigma_{e=1}^{E}(P_{\mathcal{M}_e}(y_i^*|x_i) - \hat{\mu}_i)^2}$$

We follow Swayamdipta et al. (2020) and refer to high-confidence examples as easy-to-learn (for $\mathcal{M}$), and high-variability examples as ambiguous.

2.3 Using ETD

Swayamdipta et al. (2020) have shown that training only on high-variability (or ambiguous) examples can lead to better out-of-distribution (OOD) performance, at a small cost for in-distribution (ID) performance. In this work, we show that they can be used to improve *both* ID and OOD performance.

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ be a dataset whose ETD are computed w.r.t. some model $\mathcal{M}_1$, and let $0 < f < 1, b \in \mathbb{N}$. We define $V_f(\mathcal{D})$ as the $f \cdot N$ highest-variability examples in $\mathcal{D}$. To train a model $\mathcal{M}_2$ using ETD, we construct the dataset $\mathcal{D}_{(f,b)}$: for each example $(x_i, y_i) \in \mathcal{D}$, if $(x_i, y_i) \in V_f(\mathcal{D})$, we add b copies of it to $\mathcal{D}_{(f,b)}$. Otherwise, we add 1 copy of it to $\mathcal{D}_{(f,b)}$. That is, every example from the original dataset $\mathcal{D}$ is also in $\mathcal{D}_{(f,b)}$, but $\mathcal{D}_{(f,b)}$ is **biased** towards high-variability examples by a factor of b. The new model $\mathcal{M}_2$ is then trained on $\mathcal{D}_{(f,b)}$. For example, if $f = 0.5$ and $b = 2$, $\mathcal{D}_{(f,b)}$ contains each of the top 50% highest-variability examples in $\mathcal{D}$ twice, and every other example once.

Note that this method differs from Swayamdipta et al. (2020) in that it includes all examples from the $\mathcal{D}$ in the new dataset $\mathcal{D}_{(f,b)}$. Training on all examples helps prevent the decrease in ID performance observed by Swayamdipta et al. (2020), and even improves ID performance, as we next show.

3 Training with ETD Improves Performance

We follow Swayamdipta et al. (2020) and first test the capacity of our method to improve the performance of a model of the same architecture as the model used for computing the ETD. We use six tasks, divided into three groups. Three multiple choice tasks: WinoGrande (WG; Sakaguchi et al., 2019), Abductive NLI (αNLI; Bhagavatula et al., 2019), and HellaSwag (HS; Zellers et al., 2019); Two NLI tasks: SNLI (Bowman et al., 2015) and ANLI (Nie et al., 2019); and a question answering task: BoolQ (Clark et al., 2019).

For each of the tested tasks, we compute ETD using one copy of DeBERTa-large (He et al., 2020). Following Swayamdipta et al. (2020), we use $E = 5$ as the number of iterations for the ETD computation process. We then train a second copy of DeBERTa on the ETD-weighted dataset $\mathcal{D}_{(.25,3)}$. The specific values of f and b are chosen using a grid search for the *best mean performance* over the development sets of all tasks.[3]

Due to computational constraints, we do not tune any hyperparameters other than those defined specifically for this work, i.e. f and b. For other hyperparameters such as learning-rate and batch-size we follow the values used in Swayamdipta et al. (2020) for training on the SNLI (Bowman et al., 2015) dataset.

For each task, we test the trained model on its designated test set to evaluate ID performance, as well as on the test sets of all other tasks from the same group to evaluate OOD performance (e.g., we evaluate a model trained on WinoGrande on αNLI and Hellaswag as well).

As baselines, we train DeBERTa on three additional datasets:

- $\mathcal{D}$ is the original, unaltered version of each dataset.
- $\mathcal{D}_{(.33)}$-NR is the dataset resulting from the approach of Swayamdipta et al. (2020), i.e., computing the ETD without resetting the model's weight between epochs (No Reset), and training only on 33% of the examples with the highest *variability*.
- $\mathcal{D}_{(.25,3)}$-NR is the dataset constructed by computing the ETD without resetting the model's weight between epochs (No Reset), but weighing the examples of the dataset as usual for our method.

[2] Preliminary experiments show that this method of computing confidence is better than the alternative of computing it over a separate forward-pass at the end of each iteration, so that P is the same for all examples within the iteration.

[3] Grid search is used to select all values of f and b in any method we present throughout this work. In practice, we perform a grid search using only the development sets of SNLI, ANLI, and αNLI. The development sets of the other three tasks are used for testing, as they have no accessible labeled test sets.

	SNLI		ANLI		αNLI			WinoGrande			Hellaswag			BoolQ
	ID	OOD	ID	OOD	ID	OOD		ID	OOD		ID	OOD		ID
	Test	ANLI	Test	SNLI	Test	WG	HS	Test	αNLI	HS	Test	αNLI	WG	Test
$\mathcal{D}$	91.93	32.18	55.43	85.37	86.41	64.34	68.7	84	78.13	57.13	42.69	78.58	59.36	86.97
$\mathcal{D}_{(.33)}$-NR	91.03	33.7	51.17	82.01	86.27	**65.35**	**71.5**	80.71	78.12	58.47	42.79	79.01	60.57	63.81
$\mathcal{D}_{(.25,3)}$-NR	91.79	**34.41**	54.08	84.74	87.1	64.83	66.63	84.37	**78.81**	57.31	**42.95**	78.47	61.83	86.52
Ours - $\mathcal{D}_{(.25,3)}$	**91.96**	32.91	**55.91**	**85.39**	**87.15**	64.46	70.09	**84.38**	78.32	**59.1**	42.9	**79.25**	**61.87**	**87.26**

Table 1: Training on an ETD-weighted dataset improves performance in both ID and OOD testing. Scores show the accuracy of trained models on ID and OOD test sets, compared between training on different datasets. $\mathcal{D}$ is the unaltered dataset, $\mathcal{D}_{(.33)}$-NR is the dataset from the approach of Swayamdipta et al. (2020), $\mathcal{D}_{(.25,3)}$-NR is the dataset resulting from computing the ETD without resetting the model's weight between epochs (No Reset), and **Ours** - $\mathcal{D}_{(.25,3)}$ is the dataset obtained by our approach.

	ID	OOD
SNLI	0.03	0.73
ANLI	0.48	0.02
αNLI	0.74	0.76
WinoGrande	0.38	1.08
HellaSwag	0.21	1.59
BoolQ	0.29	-
Average	0.35	0.95

Table 2: Performance improvement when training on $\mathcal{D}_{(.25,3)}$ (our approach) compared to $\mathcal{D}$ (unaltered dataset). Average is the weighted average of the scores in each column (scores are weighted by the number of tasks they represent).

Each training process is repeated with $s = 5$ different seeds, and the reported result is the mean result across seeds. For each task, we train for a fixed number of steps regardless of the size of the dataset we train on. The fixed number of steps is task-dependent, and is the number of steps required to pass $E = 5$ times on $\mathcal{D}$ and compute its ETD.

Table 1 shows the full results of this experiment. Table 2 provides a summarized version of the results, as the improvement gained by training on $\mathcal{D}_{(0.25,3)}$ compared to $\mathcal{D}$ in each task, on both ID and OOD test sets. Our method is the only one to outperform training on $\mathcal{D}$ across all 14 categories, obtaining mean improvement of 0.35% ID and 0.95% OOD. It also outperforms Dataset Cartography's approach ($\mathcal{D}_{(.33)}$-NR) on 11/14 categories, and the ablation version ($\mathcal{D}_{(.25,3)}$-NR) on 10/14 categories, demonstrating the importance of the weight reset.

These improvements are statistically significant: modelling the result of the baseline method in each category C as a sample from a normal distribution $\mathcal{N}_C$, the probability of outperforming the baseline when sampling from the same $\mathcal{N}_c$ is 0.5. Therefore, the probability of outperforming the baseline on all 14 tasks when sampling from their respective distributions can be calculated using a Binomial Random Variable $\mathcal{B}(14, 0.5)$, which gives *p-value* $\leq P[\mathcal{B}(14, 0.5) = 14] = 0.00006$.

4 ETD are Transferable

The process of computing ETD is costly in terms of compute, requiring compute roughly equivalent to that of training the desired model. Thus, computing the ETD of a dataset separately for every model we wish to train is expensive, and, depending on the size of the dataset, may become prohibitively so. This problem can be bypassed if ETD are transferable across models, i.e., if ETD can be computed using a model $\mathcal{M}_1$, and then used to train a different model $\mathcal{M}_2$.

To test for transferability, we use DeBERTa as the ETD-computing model, and ELECTRA (Clark et al., 2020) as the training model. We conduct experiments similar to those in Section 3, with the exception that the training model $\mathcal{M}_2$ is ELECTRA. Tables 3 and 4 show the results and summarized results of these experiments, respectively.

When computing ETD with DeBERTa and creating the respective $\mathcal{D}_{(0.25,4)}$, training ELECTRA on $\mathcal{D}_{(0.25,4)}$ outperforms training on $\mathcal{D}$ in 5 out of the 6 ID categories, and in 9 out of all 14 categories. Though not as consistent as our main method, the ETD transfers well, with mean improvement of 0.2% ID and 0.33% OOD.

	SNLI		ANLI		αNLI			WinoGrande			Hellaswag			BoolQ
	ID	OOD	ID	OOD	ID	OOD		ID	OOD		ID	OOD		ID
	Test	ANLI	Test	SNLI	Test	WG	HS	Test	αNLI	HS	Test	αNLI	WG	Test
$\mathcal{D}$	91.91	34.83	59.1	**83.82**	89.34	70.4	74.54	87.19	**82.02**	67.53	43.64	**81.46**	**68.27**	**87.67**
Ours - $\mathcal{D}_{(.25,4)}$	**91.96**	**35.21**	**59.7**	83.15	**89.88**	**70.73**	**76.41**	**87.21**	81.66	**70.23**	**44.61**	81.3	66.89	86.82

Table 3: Transferring ETD from DeBERTa to ELECTRA: Accuracy of trained models on ID and OOD test sets, compared between training on the unaltered dataset $\mathcal{D}$ and the ETD-weighted dataset of our approach, $\mathcal{D}_{(.25,4)}$.

	ID	OOD
SNLI	0.05	0.38
ANLI	0.6	-0.67
αNLI	0.44	1.1
WinoGrande	0.02	1.17
HellaSwag	0.97	-0.77
BoolQ	-0.85	-
Average	0.2	0.33

Table 4: Transferring ETD from DeBERTa to ELECTRA: performance improvement when training on $\mathcal{D}_{(.25,4)}$ (our approach) compared to $\mathcal{D}$ (unaltered dataset). OOD is the mean score of all OOD test sets for a given dataset. Average is the weighted average of the scores in each column (scores are weighted by the number of tasks they represent).

5 From Training Dynamics To Dynamic Training

We have established that Example Training Dynamics can be computed as a pre-processing step and used to improve both ID and OOD performance. Furthermore, thanks to tranferability it may be possible to save most of the compute involved in the process. But is it possible to go even further and use ETD without any pre-processing? After all, any training process has its own naturally-occuring Trainin Dynamics. Is it possible to compute *and* use ETD within the same training process?

With this question in mind, we propose Dynamic Training, an active learning method for weighing examples in datasets mid-training. Dynamic Training uses the same basic method as regular ETD-weighted training, except only one training process takes place, and the ETD are being evaluated during training rather than as a preprocessing step. The ETD are re-evaluated with each epoch of training, and thus the ETD-weighted dataset $\mathcal{D}_{(f,b)}$ on which the model trains changes between epochs.

A Dynamic Training process starts with no ETD whatsoever, so training is done on the unaltered dataset $\mathcal{D}$. Similar to computing regular ETD, as training progresses, we save the model's probabilities for the labels of each example in $\mathcal{D}$ as ETD. On some epoch $e^* > 1$, we start using them to weigh $\mathcal{D}$ before each epoch. From e^* onwards, we train each epoch on $\mathcal{D}_{(f,b)}$. We keep saving the model's output probabilities to update the ETD, and before the start of each epoch, calculate a new $\mathcal{D}_{(f,b)}$. As in the previous experiments, training is set to a fixed number of steps for each task, regardless of the varying size of $\mathcal{D}_{(f,b)}$ during training. Fig. 2 illustrates the flow of a Dynamic Training process.

Experiments To test the effectiveness of Dynamic Training , we use the method to train a DeBERTa model on the six datasets we used for our previous experiments. We use the hyperparameters $e^* = 3, f = 0.33, b = 2$ (based on a grid search for the best performance on the development sets). We compare the results against a DeBERTa model trained on $\mathcal{D}$ without any form of ETD-weighted Training. Tables 5 and 6 show the results and summarized results of these experiments.

Dynamic Training outperforms training on $\mathcal{D}$ in 5 out of the 6 ID categories, and in 11 out of all 14 categories. Though the ANLI task suffers decrease in performance, results for the other tasks improve relatively consistently, with mean improvement of 0.23% ID and 0.38% OOD. Though not as effective as ETD-weighted training, Dynamic Training improves performance in the majority of the cases, without any pre-processing of the data.

6 Robustness to Hyperparameter tuning

Throughout this work we report results training models on ETD-weighted datasets of the form $\mathcal{D}_{(f,b)}$, with specific values chosen for f, b. These values are hyperparameters, chosen using a grid search for best performance over the development set of the different tasks. Table 7 shows the other values of f, b we tested for in Section 3, and their

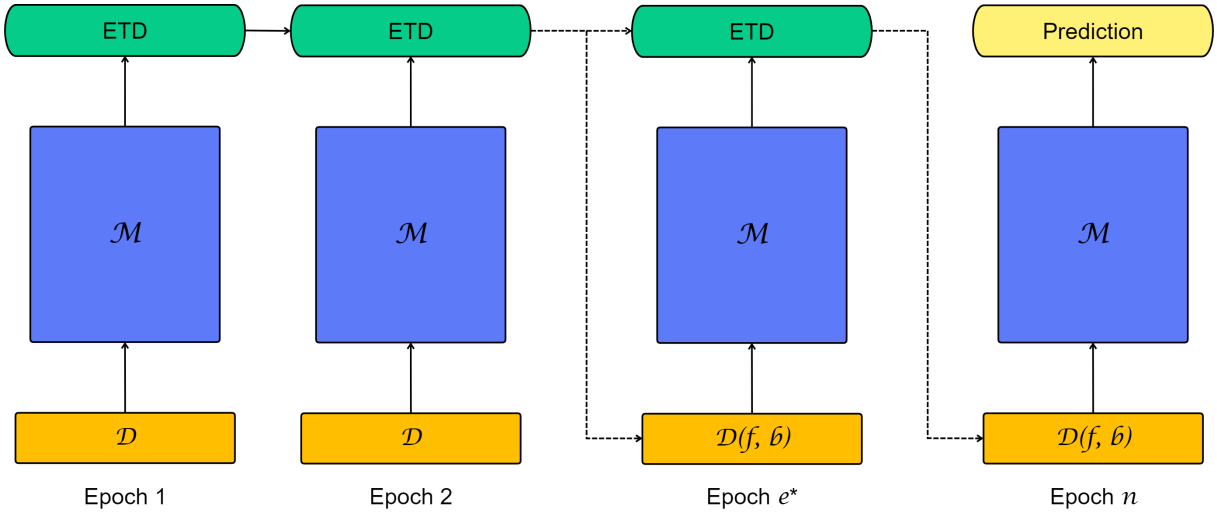

Figure 2: Dynamic Training of ETD. $\mathcal{M}$ is trained on $\mathcal{D}$, accumulating ETD with each epoch. Starting at epoch e^* and on each epoch afterwards, $\mathcal{M}$ is trained on $\mathcal{D}_{(f,b)}$ instead of $\mathcal{D}$. The ETD and $\mathcal{D}_{(f,b)}$ keep updating until the end of training.

	SNLI		ANLI		αNLI			WinoGrande			Hellaswag			BoolQ
	ID	OOD	ID	OOD	ID	OOD		ID	OOD		ID	OOD		ID
	Test	ANLI	Test	SNLI	Test	WG	HS	Test	αNLI	HS	Test	αNLI	WG	Test
$\mathcal{D}$	91.93	32.18	**55.43**	**85.37**	86.41	64.34	68.7	84	78.13	**57.13**	42.69	78.58	59.36	86.97
DT - $\mathcal{D}_{(.33,2)}$	**92.04**	**32.26**	55.38	85.15	**86.91**	**65.06**	**69.42**	**84.92**	**78.73**	57.12	**43.01**	**78.69**	**60.36**	**86.99**

Table 5: Dynamic Training results. $\mathcal{D}$ is training on the unaltered dataset, and **DT** - $\mathcal{D}_{(.33,2)}$ is Dynamic Training on $\mathcal{D}_{(.33,2)}$ starting at epoch $e^* = 3$.

	ID	OOD
SNLI	0.11	0.08
ANLI	-0.05	-0.22
αNLI	0.5	0.72
WinoGrande	0.52	0.3
HellaSwag	0.32	0.56
BoolQ	0.02	-
Average	0.23	0.38

Table 6: Performance improvement when using Dynamic Training on $\mathcal{D}_{(.33,2)}$ starting at epoch $e^* = 3$, compared to training on $\mathcal{D}$ without any form of ETD-weighted Training. OOD is the mean score of all OOD test sets for a given dataset. Average is the weighted average of the scores in each column (scores are weighted by the number of tasks they represent).

model's respective improvement in performance. All values for f, b improve performance over the baseline in at least 8 out of 11 tests, with 85/99 of all test scores being positive, implying our method is robust with respect to its hyperparameters.

7 Related Work

Previous research offered various methods that consider the relative importance of examples within a dataset in order to improve training. Methods such as Curriculum learning (Bengio et al., 2009) or self-paced learning (Kumar et al., 2010) focus on the order of training, advocating training on easy examples first (e.g., examples with high likelihood in the latter case).

Other methods rank the importance of examples with the goal of filtering datasets. Liu et al. (2021) use a train-twice approach with some resemblance to ETD, and upweight train examples that were missclassified in the first round of training. AFLite (Le Bras et al., 2020) ranks examples based on the ability of a linear classifier to solve them, and then filters out easy examples in order to eliminate artifacts and biases from the dataset. Other works advocated bias and artifacts removal, and by extension the removal of easy examples from datasets (Gururangan et al., 2018; Li and Vasconcelos, 2019).

Several approaches have used training metrics

	SNLI		ANLI		αNLI		WinoGrande		Hellaswag		BoolQ
	ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID
$\mathcal{D}_{(.25,2)}$	-0.02	-0.48	**0.42**	**0.34**	**0.68**	**0.32**	**0.02**	**2.17**	**0.35**	**0.96**	**0.48**
$\mathcal{D}_{(.33,2)}$	**0.12**	**1.48**	-0.48	**0.03**	**0.61**	**0.57**	**0.43**	**1.14**	**0.36**	**1.58**	**0.04**
$\mathcal{D}_{(.50,2)}$	**0.11**	**0.65**	**0.07**	**0.04**	**0.88**	-1.23	-0.08	**1.65**	**0.27**	**0.35**	-0.05
$\mathcal{D}_{(.25,3)}$	**0.03**	**0.73**	**0.48**	**0.02**	**0.74**	**0.76**	**0.38**	**1.08**	**0.21**	**1.59**	**0.29**
$\mathcal{D}_{(.33,3)}$	**0.10**	**1.23**	-0.40	**0.52**	**0.95**	**0.57**	**0.97**	**1.03**	**0.41**	**1.52**	**0.07**
$\mathcal{D}_{(.50,3)}$	**0.11**	**0.58**	-0.62	**0.11**	**0.61**	**1.16**	**0.08**	**1.52**	**0.22**	**0.37**	**0.17**
$\mathcal{D}_{(.25,4)}$	**0.15**	-0.18	-0.72	**0.45**	**0.89**	-0.48	**0.54**	**1.67**	**0.45**	**0.95**	**0.06**
$\mathcal{D}_{(.33,4)}$	**0.09**	**0.82**	-0.33	**0.04**	**0.69**	**0.94**	**0.32**	**1.10**	**0.50**	**1.07**	**0.14**
$\mathcal{D}_{(.50,4)}$	**0.02**	**0.48**	-0.35	-0.05	**0.70**	**1.27**	-0.01	**0.89**	**0.27**	**1.59**	**0.26**

Table 7: Performance improvement when training DeBERTa-large on an ETD-weighted dataset (ETD computed with DeBERTa-large) over training without ETD, compared between different values for f, b. OOD is the mean score of all OOD development sets. Positive scores are marked in bold font

such as training loss (Han et al., 2018; Arazo et al., 2019; Shen and Sanghavi, 2018) and confidence (Hovy et al., 2013) to filter mislabeled examples out of datasets. Chang et al. (2017) used the training metrics of confidence and variability to reweigh the examples of a dataset by importance after each epoch of training, similar to the method we propose in Section 5 of this work. However, they focus on simple vision-related tasks and CNNs, and don't report results for OOD generalization. Wang et al. (2019) use a dynamic sampling method based on the metric of training-cost difference between two consecutive epochs to achieve improved results and accelerated training on the task of machine translation. Toneva et al. (2019) defined the forgettability metric, which measures how often an example is forgotten during training, and is similar to our notion of variability. They showed that training on a subset of forgettable examples can still maintain ID testing results, but did not report ID performance improvement. They also tested the transferability of the forgettability metric between models. Similarly to Chang et al. (2017), they focused on simple vision-related tasks and didn't report OOD results.

Core-set selection (Wei et al., 2013) uses submodular functions to find a subset of a dataset representative of the whole, to be used under low computational budget. Conversely, our approach uses training metrics to find a subset not necessarily representative of the whole, but rather one that can be emphasized within the whole dataset to improve performance, regardless of budget considerations. Similarities can also be drawn between our approach and active learning (Settles, 2009; Peris and Casacuberta, 2018; P.V.S and Meyer, 2019), which searches unlabeled data for the most useful examples to label and uses them for training.

Sanh et al. (2020) aim to remove biases from models without re-sampling of the dataset, using Product of Experts between a weak (biased) model and a main model and achieving improvements over OOD test-sets. Karimi Mahabadi et al. (2020) suggested a somewhat similar approach of de-biasing a main model by contrasting it with a "bias-only" model, to achieve OOD improvements in tasks where the bias is known. Nam et al. (2020) also used biased models as a foil to a main model, to achieve de-biasing in vision-related tasks. Utama et al. (2020) proposed a debiasing method that improves OOD testing while maintaining ID test results. Their method regulates the model's confidence over biased examples in the dataset, using knowledge distillation in combination with biased models. This approach requires a-priori knowledge of the dataset's biases in order to formulate the biased model, and as such is not applicable to many NLP tasks. It also requires a large amount of compute, as it trains a full-size teacher model and a biased model besides the main model.

8 Conclusion

We presented a new method for computing Example Training Dynamics, which can be used to increase both ID and OOD performance, without any changes to model size or architecture. We demonstrated that ETD can be transferable, i.e., they can be computed once and used many times for different models, reducing the computation cost at a long term. Finally, we have shown that ETD can be computed on the fly using Dynamic Training, which may hold the key to improved performance using ETD at no extra compute cost.

As the field of NLP leans more and more into

the self-supervised pre-training paradigm, further research on ETD may be focused on adjusting our method for larger and self-supervised datasets in order to improve and reduce the cost of pre-training as a whole.

Acknowledgments

We thank Omer Levy for his insightful comments that contributed to this paper. We also thank our anonymous reviewers for their constructive feedback. This work was supported in part by the Israel Science Foundation (grant no. 2045/21).

Limitations

While this work covers as many domains as its computational budget allows, it still only tests for tasks in the English language, and only for NLI, multiple-choice and question-answering tasks. Our approach requires a labeled set of examples to be applied, so that adapting it to other settings, specifically pre-training, is not straightforward.

The results presented in this work are not always consistent across all tasks. While our main approach improves performance consistently, the methods of Transferability and Dynamic Training are not as consistent and result in decrease in performance for some test-cases. The relation between method and task should be further researched, so that future application of our approach can match the right method to a dataset.

Our approach does not have any clear risks or negative societal impacts. We hypothesize it might have the positive impact of social debiasing, as emphasizing *ambiguous* examples can be considered a form of debiasing in the case that the majority of examples are biased towards a specific gender or race. Testing for this is deferred to future work.

References

Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. 2019. Unsupervised label noise modeling and loss correction. arXiv:1904.11238.

Y. Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, volume 60, page 6.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2019. Abductive commonsense reasoning. arXiv:1908.05739.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. 2017. Active bias: Training more accurate neural networks by emphasizing high variance samples. arXiv:1704.07433.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. arXiv:1905.10044.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. arXiv:2003.10555.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. arXiv:1804.06872.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. arXiv:2006.03654.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. arXiv:2203.15556.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

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.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

M. Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E. Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. arXiv:2002.04108.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The Winograd Schema Challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'12, pages 552–561. AAAI Press, Rome, Italy.

Yi Li and Nuno Vasconcelos. 2019. Repair: Removing representation bias by dataset resampling. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9564–9573.

Evan Zheran 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. arXiv:2107.09044.

Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. 2020. Learning from failure: Training debiased classifier from biased classifier. arXiv:2007.02561.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. arXiv:1910.14599.

Álvaro Peris and Francisco Casacuberta. 2018. Active learning for interactive neural machine translation of data streams. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 151–160, Brussels, Belgium. Association for Computational Linguistics.

Avinesh P.V.S and Christian M. Meyer. 2019. Data-efficient neural text compression with interactive learning. 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 2543–2554, Minneapolis, Minnesota. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. WINOGRANDE: an adversarial winograd schema challenge at scale. *CoRR*, abs/1907.10641. arXiv:1907.10641.

Victor Sanh, Thomas Wolf, Yonatan Belinkov, and Alexander M. Rush. 2020. Learning from others' mistakes: Avoiding dataset biases without modeling them. arXiv:2012.01300.

Burr Settles. 2009. Active learning literature survey.

Yanyao Shen and Sujay Sanghavi. 2018. Learning with bad training data via iterative trimmed loss minimization. arXiv:1810.11874.

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*.

Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020. Mind the trade-off: Debiasing nlu models without degrading the in-distribution performance. arXiv:2005.00315.

Rui Wang, Masao Utiyama, and Eiichiro Sumita. 2019. Dynamic sentence sampling for efficient training of neural machine translation. arXiv:1805.00178.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. 2013. Using document summarization techniques for speech data subset selection. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 721–726, Atlanta, Georgia. 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. 2019. Huggingface's transformers: State-of-the-art natural language processing. arXiv:1910.03771.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? arXiv:1905.07830.

A Datasets

We use the following six datasets to test our methods. Note that BoolQ, HellaSwag and WinoGrande don't have published labeled test sets (See Section 3).

αNLI (Bhagavatula et al., 2019) is a dataset testing for inference to the most plausible explanation. Given two observation-sentences, the task is to choose the best of two hypothesis-sentences to fit in-between the observation-sentences to create a plausible narrative. It applies adversarial filtering to remove artificial biases. Its train/dev/test split is 169654/1532/3069.

Adversarial NLI (ANLI) (Nie et al., 2019) is a dataset curated by an adversarial human-and-model-in-the-loop method. Starting with a base model trained on an NLI task, human annotators provide new examples that are misclassified by the model. A new model is then trained on those examples, and the process repeats. We use the version of the dataset from the third (and last) published round of this process. Its train/dev/test split is 100459/1200/1200.

BoolQ (Clark et al., 2019) is a dataset of naturally-occuring yes/no questions collected from queries to the Google search engine. Its train/dev/test split is 9427/3270/3245.

HellaSwag (Zellers et al., 2019) is a dataset testing for commonsense-NLI. Given a premise, the task is to choose the most likely hypothesis to continue the premise out of four options. HellaSwag uses Adversarial filtering to remove artificial biases and create a more challenging dataset. Its train/dev/test split is 39905/10042/10003.

Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) is a large corpus of NLI examples. Its train/dev/test split is 550152/10000/10000. Our preliminary experiments showed no significant difference between training DeBERTa-large and ELECTRA-large models on the full SNLI train-set, and on 10% of the train-set. To save time and compute, we use only the first 10% of the SNLI train-set examples for all experiments in this work.

WinoGrande (Sakaguchi et al., 2019) is a large-scale dataset of pronoun-resolution problems inspired by the Winograd Schema Challenge (WSC; Levesque et al. (2012)) design. It applies adversarial filtering to remove artificial biases. Its train/dev/test split is 40398/1267/1767.

B Experimental Settings

In all our experiments, we minimize cross entropy with the Adam optimizer (Kingma and Ba, 2014), following the AdamW learning rate schedule from the PyTorch library. We use a batch size of 128 in all experiments. Each experiment is run with 5 random seeds.

αNLI and ANLI model train for 4800 optimization steps each. SNLI models are trained for 2600 steps. Winogrande, Hellaswag and BoolQ models are trained for 2400 steps. The difference in steps serves to adjust the training time to the size of the training set, to allow each baseline model to train fot 5 epochs, following (Swayamdipta et al., 2020)

We train on RTX208, Quadro RTX 600 and A10 GPU's. We estimate our training at an average of 8 hours per experiment, and approximately 9120 GPU hours in total.

Our implementation uses the Huggingface Transformers library (Wolf et al., 2019).

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
10

☑ A2. Did you discuss any potential risks of your work?
10

☑ A3. Do the abstract and introduction summarize the paper's main claims?
abstract, section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

section 4, appendix A

☑ B1. Did you cite the creators of artifacts you used?
Left blank.

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We use 6 well-known datasets dor their intended use of academic research.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We use 6 well-known datasets dor their intended use of academic research.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We did not collect any data ourselves. only used datasets published by others in previous works. It is artificial data and does not contain any personal information.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
section 10, Appendix A

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
appendix A

C ☑ Did you run computational experiments?

sections 4, 5, 6, appendix B

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
*appendix B *Except the number of parameters. we only use DeBERTa-large and ELECTRA-large models, for which the number of parameters is well documented.*

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Sections 4, 7, appendix B

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Sections 4, 5, 6

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
appendix B

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.