Dynamic Augmentation Data Selection for Few-shot Text Classification

Guangliang Liu
Michigan State University
liuguan5@msu.edu

Lifeng Jin
Tencent AI Lab
lifeng.jin@gmail.com

Owen Yuan
Lynbrook High School
oyuan688@student.fuhsd.org

Jiayu Zhou
Michigan State University
jiayuz@msu.edu

Abstract

Data augmentation has been a popular method for fine-tuning pre-trained language models to increase model robustness and performance. With augmentation data coming from modifying gold train data (in-sample augmentation) or being harvested from general domain unlabeled data (out-of-sample augmentation), the quality of such data is the key to successful fine-tuning. In this paper, we propose a dynamic data selection method[1] to select effective augmentation data from different augmentation sources according to the model's learning stage, by identifying a set of augmentation samples that optimally facilitates the learning process of the most current model. The method firstly filters out augmentation samples with noisy pseudo labels through a curriculum learning strategy, then estimates the effectiveness of reserved augmentation data by its influence scores on the current model at every update, allowing the data selection process tightly tailored to model parameters. And the two-stage augmentation strategy considers in-sample augmentation and out-of-sample augmentation in different learning stages. Experiments with both kinds of augmentation data on a variety of sentence classification tasks show that our method outperforms strong baselines, proving the effectiveness of our method. Analysis confirms the dynamic nature of the data effectiveness and the importance of model learning stages in utilization of augmentation data.

1 Introduction

Finetuning pre-trained language models (PLMs), such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019), requires a large amount of high-quality labeled data to reach high performance. Recently, few-shot learning with PLMs (Gao et al., 2021; Perez et al., 2021) has become a popular research topic, where approaches have been proposed to help fine-tuned PLMs achieve good performance in tasks where gold training data is scarce. One significant challenge in fine-tuning PLMs in low-resource settings is that the model can easily overfit to the training data of a small size and have a hard time generalizing to unseen data (Jiang et al., 2019; Zhou et al., 2021). Data augmentation (DA) is a popular method in preventing PLMs from overfitting by creating new training data by either modifying existing training samples with external resources such as a machine translation engine, or finding new training samples with pseudo-labels using tools and heuristics (Yang et al., 2020). DA has been successfully applied to various tasks such as image classification (Perez and Wang, 2017; Shorten and Khoshgoftaar, 2019), object detection (Zoph et al., 2020), text classification (Sun et al., 2019; Wu et al., 2022), and neural machine translation (Wei et al., 2022, 2020; Gao et al., 2019; Cheng et al., 2020) to improve model performance.

The quality of augmentation data is crucial to the success of the application of DA to a task, therefore data selection (e.g. Feng et al., 2021; Mou et al., 2021; Hong et al., 2022) is necessary where augmentation instances are chosen for their effectiveness in improving model performance on the task. This is especially true for silver training instances harvested from general domain sources such as Wikipedia. Such silver instances can be used to enhance model generalizability due to their out-of-sample nature, whereas data augmentation through data modification (Wei and Zou, 2019; Sennrich et al., 2016) has limited capability. However, out-of-sample augmentation training instances tend to be noisier and more numerous than in-sample augmentation instances, exacerbating the problem of selecting the effective augmentation instances (Mou et al., 2021; Qu et al., 2020).

For both kinds of augmentation data, the effectiveness for improving model performance is highly

[1]Our code is available at https://github.com/illidanlab/DynSelector

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 4870 - 4881

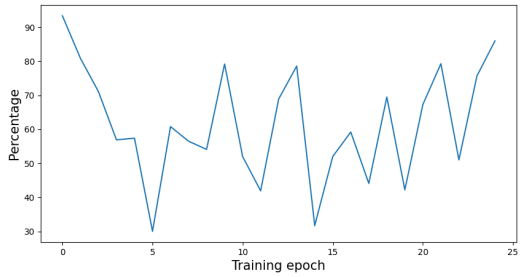

Figure 1: Percentage of effective augmentation instances after each training epoch. Before training, 1,000 effective augmentation instances, which can decrease the test error, are chosen. The percentage of the chosen augmentation instances being still effective after each epoch of training varies widely from around 80% at epoch 24 to close to 30% at epoch 5, indicating that the effectiveness of augmentation data is linked to the model learning stage.

dynamic and changes as training progresses. This can be observed in Figure 1 where 1,000 silver instances are chosen for their oracle effectiveness in reducing the test error for a stance detection task (Mohammad et al., 2016) with a randomly initialized model. After training with gold data for a different number of epochs, the number of effective augmentation data among the 1,000 instances changes drastically, showcasing how the effectiveness of the silver instances changes dynamically at different learning stages. However, previous research either does not pay attention to the interaction between the learning stage of the model and silver instance effectiveness (Yang et al., 2020), or only considers representation similarity dynamically, such as choosing augmentation samples generated from the current batch (Mou et al., 2021; Qu et al., 2020; Xie et al., 2020) while missing the arguably more important aspect of dynamic effectiveness.

In this paper, we address the augmentation instance selection problem for both in-sample and out-of-sample data augmentation with a unified framework centered around data effectiveness. The framework, named DynSelector, leverages the influence function (IF, Koh and Liang, 2017) to directly estimate the effectiveness of each augmentation sample dynamically based on the current learning stage. This ensures the selected augmentation samples always have a positive influence on model performance. When coupled with a curriculum learning-based strategy for mitigating the noisy supervision from augmentation data, our augmentation data selection method can effectively utilize and combine different augmentation sources to improve few-shot text classification performance.

2 Related Work

Data Augmentation for NLP. From the perspective of semantic consistency, there are two main research lines of DA approaches: label-preserving augmentation and label-mixing augmentation (Wu et al., 2020). Label-preserving methods synthesize augmentation data that has the same label with the referring instance, such as EDA (Wei and Zou, 2019) and BACKTRANSLATION (Sennrich et al., 2016). EDA applies random perturbation operations on the input space of the referred training instance, such as insertion, swap, synonym replacement and deletion. BackTrans(BackTrans) is a model-based augmentation method, it firstly translates a sequence of tokens into another language and then translates it back to the original language. Another research line of DA is based on MIXUP (Zhang et al., 2018) and it interpolates the inputs and labels of several training samples, in order to create semantically inconsistent augmentation samples (Jindal et al., 2020; Guo, 2020). Most existing data augmentation research creates new samples based on the train data, and we propose to leverage information from the general domains.

Data Selection is a common problem in a lot of learning tasks, such as semi-supervised learning (Zhang and Rudnicky, 2006; Zhang et al., 2021), co-training (Chen et al., 2011), learning from noisy supervision (Shu et al., 2019; Sun et al., 2020) and active learning (Fu et al., 2013). The goal of data selection is to filter out unacceptable data or re-weight the given data according to some specific purpose. Shu et al. (2019) learns a two-layered neural network to assign lower weights to noisy data and higher weights to clean data. Wang et al. (2021) applies a meta-learning framework to re-weight the pseudo-labeled data to expand training data size for few-shot sequence labeling tasks. Wang et al. (2020) employs a reinforcement learning-based data selector to select samples, with better generalization capacity, from the training data. And the selection decision is determined by the gradient similarity between a train sample and the validation dataset.

When it comes to augmentation data selection, Zhou et al. (2021) empirically proves that the main bottleneck of data augmentation for few-shot learning comes from the corrupted label in the augmen-

tation data. Mou et al. (2021) applies contrastive learning to enforce augmentation data with various weights to be separable, according to the assigned weight, in the embedding space. Yang et al. (2020) implements two filters to detect effective augmentation samples from data candidates generated by a language generative model. One of the filter is based on influence function to evaluate the change of generalization error if accepting one augmentation sample and another filter evaluates the diversity property of augmentation data. Our method leverages the influence function to estimate the effectiveness of augmentation data, and only considers augmentation data that has a positive influence to the model's performance.

3 Method

We propose an algorithm which dynamically selects augmentation instances based on the effectiveness on a model learning stage from pools of silver training instances generated by in-sample or out-of-sample augmentation methods. At a high level, the algorithm selects a batch of augmentation instances after each model update with gold training instances, based on the influence scores of the augmentation instances to ensure their effectiveness for the most current model. When both in-sample and out-of-sample silver instances are available, the algorithm first selects in-sample silver instances to ensure the model is well-trained, and then expands into out-of-sample silver instances to boost the model's generalizability and robustness once the model has approached convergence. We specify in detail each part of the algorithm below.

3.1 Dynamic augmentation data selection

Let θ be the parameter of the classification model f, the training data $\mathrm{D}_{\mathrm{train}}$, validation data $\mathrm{D}_{\mathrm{valid}}$, and augmentation data $\mathrm{D}_{\mathrm{aug}}$. An individual sample is represented with (x, y) where x is the input data, y is the associated label. $L(\cdot; \theta)$ is the classification loss function, such as cross-entropy loss.

Algorithm 1 shows the pseudo code of our method. At the beginning of each training epoch t, we first calculate the prediction probability threshold with equation 2 of the curriculum learning (CL) strategy (Section 3.3), which is a weighted maximum pseudo-label prediction probability threshold from all silver instances used in the previous epoch. This threshold is employed to filter out augmentation instances with low prediction probabilities

Algorithm 1: The Dynamic Augmentation Data Selection Method (DynSelector)

Input: the classification model f_θ, augmentation data budget $r * |D_{\mathrm{train}}|(r \geq 0)$, prediction probability threshold on the t-th training epoch τ_t, maximum training epoch T and maximum training iterations per epoch B

```
1  while t < T do
2      get τ_t with equation 2, and set n_aug = 0
3      while b < B do
4          sample a mini-batch b_train ~ D_train
5          update θ with L(b_train; θ)
6          if n_aug >= r * |D_train| then
7              continue
8          else
9              Sampling (x, y) ~ D_aug
10             if p(y|f_θ(x)) < τ_t then
11                 continue
12             else
13                 get IF(x, y) with equation 1
14                 if IF(x, y) < 0 then
15                     update θ with L(x, y; θ)
16                     n_aug ++
17                 end
18             end
19         end
20         update θ with L(b_train; θ)
21     end
22 end
```

at the current epoch. In addition, a counter n_{aug} is initialized to count how many augmentation update steps have been finished. If the counter has met the augmentation budget $r * |D_{\mathrm{train}}|$, then the augmentation operations would be blocked. The augmentation budget constrains the contribution of augmentation data and it varies by data quality. For instance, given the high-quality augmentation data acquired with BackTrans, the budget ratio can be 1 or larger.

For lines 4 - 5, the model is updated with a mini-batch gold training data b_{train}. After that, an augmentation data-point (x, y) is sampled from an augmentation data pool D_{aug}. If the model's prediction probability on its pseudo-label $p(y|f_\theta(x))$ is smaller than the threshold τ_t from line 2, this data-point would be rejected. However, if the augmentation data is accepted, its influence score with

regard to the current model would be calculated as shown in line 13. The influence score indicates how the test error might change if this augmentation sample is adopted to train the model. Therefore augmentation samples with large prediction probabilities and negative influence scores would be used to update the classification model at this step. Influence value estimation is described in Section 3.2. At the end of each epoch, the gold training mini-batch data would be reused to update the model again (line 20), in order to mitigate the impact of noisy augmentation brought by the IF value estimation error.

We consider different utilization strategies for in-sample and out-of-sample augmentation. Given out-of-sample augmentation data, we ask the model to be well-trained with gold data first. When both in-sample and out-of-sample augmentation data are available, we use the in-sample augmentation data first with our algorithm. As long as the model has converged, we train it further with the out-of-sample augmentation. The two-stage augmentation case is described in Section 3.4.

3.2 IF Value Estimation

Influence function is first introduced to deep learning by (Koh and Liang, 2017), in order to estimate the influence of perturbing a training sample to the prediction of a test sample. Yang et al. (2020) extends it to evaluate the generalization ability of augmentation samples. Given a classification model f_θ, the influence of an augmentation sample $\mathrm{IF}(x_{\mathrm{aug}}, y_{\mathrm{aug}})$ to the prediction loss of validation data D_{valid} is defined as:

$$L(D_{\mathrm{valid}}; f_\theta(D_{\mathrm{train}} \cup (x_{\mathrm{aug}}, y_{\mathrm{aug}}))) \\ -L(D_{\mathrm{valid}}; f_\theta(D_{\mathrm{train}}))$$

where the validation loss approximates the generalization loss. $\mathrm{IF}(x_{\mathrm{aug}}, y_{\mathrm{aug}}) < 0$ indicates that adopting the sample $(x_{\mathrm{aug}}, y_{\mathrm{aug}})$ to update θ decreases validation loss, and therefore we know that combining the augmentation sample with the training data to train the classification model would improve the model's generalization ability. Following Koh and Liang (2017), $\mathrm{IF}(x_{\mathrm{aug}}, y_{\mathrm{aug}})$ can be approximated with:

$$\frac{-1}{|D_{\mathrm{valid}}|} \nabla_\theta L(D_{\mathrm{valid}}; \theta_t)^T H_{\theta_t}^{-1} \nabla_\theta L(x_{\mathrm{aug}}, y_{\mathrm{aug}}; \theta_t). \quad (1)$$

At the training iteration t, given the up-to-date parameter θ_t, calculating equation 1 is computationally prohibitive because of the inverse hessian matrix $H_{\theta_t}^{-1}$. We adopt the stochastic estimation proposed by Koh and Liang (2017) to estimate $H_{\theta_t}^{-1} \nabla_\theta L(D_{\mathrm{valid}}; \theta_t)$. Comparing to gradient similarity calculation between a given data point and the validation data (Ren et al., 2018; Wang et al., 2020, 2021) used in meta-learning, the IF calculation in Equation 1 measures the similarity between augmentation data gradient and validation data gradient projected into the space of Hessian matrix, which indicates the optimal gradient descent direction.

Because we need to calculate the inverse Hessian matrix every time the model is updated, which happens between any two connective training iterations, stochastic estimation at each training iteration is computationally prohibitive. We propose to use Sherman-Morrison formula (Hager, 1989) to efficiently calculate the inverse hessian matrix. Specifically, for any two connective training iterations t and $t+1$ and the associated model parameters θ_t and θ_{t+1}, the inverse hessian matrix $H_{\theta_{t+1}}^{-1}$ can be computed by:

$$H_{\theta_{t+1}}^{-1} = H_{\theta_t}^{-1} + \\ \frac{(\triangle J_{t+1} - H_{\theta_t}^{-1} \triangle \theta_{t+1})(\triangle J_{t+1} - H_{\theta_t}^{-1} \triangle \theta_{t+1})^T}{(\triangle J_{t+1} - H_{\theta_t}^{-1} \triangle \theta_{t+1})^T \triangle \theta_{t+1}}$$

$$\triangle J_{t+1} = \nabla_\theta L(\cdot; \theta_{t+1}) - \nabla_\theta L(\cdot; \theta_t)$$

$$\triangle \theta_{t+1} = \theta_{t+1} - \theta_t$$

The stochastic estimation of $H_{\theta_t}^{-1} \nabla_\theta L(\mathrm{D}_{\mathrm{valid}}; \theta_t)$ requires us to calculate $\nabla_\theta^{-1} L(\mathrm{D}_{\mathrm{valid};\theta_t})$ to get $H_{\theta_{t+1}}^{-1}$ with Sherman-Morrison formula. However, the Jacobian matrix of θ is not always square, therefore we calculate the pseudo-inverse matrix of $\nabla_\theta L(\mathrm{D}_{\mathrm{valid}}; \theta_t)$. Besides the inverse Jacobian matrix, acquiring the gradient w.r.t validation data and the augmentation data respectively is expensive as well. In section 5.2, we conduct ablation studies to verify the relation between classification performance and the frequency of inverse Hessian matrix update.

3.3 Curriculum Learning Strategy

The curriculum learning (CL) strategy rejects noisy augmentation data and reduces the search space of data candidates. We concern more about the quality of augmentation data than the performance of learning models, so our method makes a selection decision based on the prediction probability on the pseudo label instead of the maximum prediction probability over all labels (Sohn et al., 2020; Zhang et al., 2021).

Given a pre-defined ratio $\tau \in (0, 1)$ and associated decreasing step η, a sub-set $D^{'}$ sampled from D_{aug}, the prediction probability threshold at the training epoch t is:

$$\tau_t = \max\left(\frac{1}{l}, \tau_t^{'}\right) \tag{2}$$

$$\tau_t^{'} = (\tau - \eta * (t-1)) * \underset{(x_i, y_i) \in D^{'}}{\arg\max}\, p(y_i | f_\theta(x_i)),$$

where $t \in \{1, 2, ..., \mathrm{T}\}$ and $\eta \in (0, 1)$, l is the label space size, η controls the step to expand the search space of augmentation data. Since out-of-sample augmentation instances are more noisy, we set η to zero for out-of-sample data. Provided a large τ, the CL strategy always select the highest quality out-of-sample data in the course of training. In contrast, the η for in-sample instances is positive, in order to consider an increasingly number of augmentation data as the training process goes on. For the purpose of computational efficiency, the CL strategy works in the beginning of each training epoch as shown in Algorithm 1, but it can be easily extended to a mini-batch manner.

3.4 Two-stage Augmentation

Both in-sample and out-of-sample augmentation data can improve PLMs' performance on few-shot learning tasks through providing more training data, however the motivation behind them are different. In-sample augmentation data regularizes the learning process with more in-sample data and prevents few-shot learning models from overfitting to training data. The out-of-sample data is more likely to contain novel information, which enhances the generalization and robustness of learning models. Therefore, they should be used in different ways, especially when both are available for a task.

We take the advantages of both in-sample augmentation data and out-of-sample augmentation data with our dynamic selection algorithm in a two-stage augmentation paradigm. We first use the in-sample augmentation data together with the gold training data for training until convergence. This allows the trained model to achieve good performance in in-sample evaluation. Once the model has met the early stop criteria, our algorithm takes the out-of-sample augmentation data to maximize the generalization ability of the model.

4 Experiments

We conduct extensive experiments with the proposed algorithm and a variety of classification tasks. We adopt BERT (Devlin et al., 2018) as our base modeland one fully-connected layer as the classifier for all experiments, with details about the training hyperparameters described in Appendix A.1. All the models are implemented with PyTorch (Paszke et al., 2017).

4.1 Setup

We conduct experiments with three different augmentation data scenarios based on augmentation data availability: both in-sample and out-of-sample data available, only out-of-sample data available and only in-sample data available. For the first two scenarios, the tasks are stance detection and sentiment analysis. For the last scenario, the experiments are conducted on the above two tasks as well as six tasks from the GLUE benchmark.

To simulate few-shot settings, we randomly extract 220 samples from the original training set, and 200 of them work as training data and the rest is used as validation data. We repeat all experiments over 5 random seeds and report the average performance on the original development sets.

4.2 Dataset

We conduct two-stage augmentation and out-of-sample augmentation data experiments on the stance detection task (SD) from the SemEval 2016 stance dataset (Mohammad et al., 2016) and the Chinese sentiment analysis task(CNSA) from SMP2020-EWECT[2]. We also test the performance of our method on the in-sample augmentation over 6 tasks out of the GLUE benchmark: CoLA, SST-2, QQP, MNLI-m, QNLI, and RTE.

In terms of in-sample data augmentation, we implement two representative text data augmentation methods: BackTrans (Xie et al., 2020) and EDA (Wei and Zou, 2019). For datasets with paired data inputs (QQP, MNLI-m, QNLI, RTE), we generated 16 total augmentation data samples with EDA and 2 augmentation data samples with BackTrans - half of each were generated on the first part of the input, and the other half was generated using the second part of the paired data input. For example, for QQP, we generated 8 EDA augmentation data and 1 BackTrans augmentation data by modifying the first question and generated the other augmentation data by running EDA and BackTrans on the second question. For the other datasets (CoLA, SST-2, CNSA, SD), we generated

[2]https://smp2020ewect.github.io/

16 augmentation samples for each training instance with EDA and 1 augmentation sample with Back-Trans. The out-of-sample augmentation data for stance classification is collected from Wikipedia by labeling consecutive sentences with a causal relation as having a support relation, and sentences with a contradiction relation as having an against relation. For sentiment analysis, the out-of-sample data is collected by mapping emojis in tweets to one of five sentiments in target set of the task. We randomly sampled 100,000 out-of-sample augmentation instances from the collected data source.

4.3 Baseline Methods

We consider 9 baseline models. All baseline models are trained the same augmentation data if augmentation is used.

- BERT-base without augmentation (BERT).
- BERT-base with mixed dataset of training data and subsets sampled from augmentation data at each training epoch (BERT-mix).
- BERT-base with randomly sampled augmentation data(BERT-RandomAug).
- BERT-base with augmentation data selected by gradient similarity between validation data and augmentation data (BERT-GradSim). Following Yu et al. (2020), we define gradient similarity as the cosine similarity between gradient vector w.r.t validation data and the gradient vector w.r.t an augmentation sample.
- BERT-base with augmentation data selected by IF value only without CL (Ours-noCL).
- BERT-base with augmentation data selected by the CL strategy alone (Ours-noIF).
- Following Yang et al. (2020) where all augmentation data should be fed into the influence value filter firstly, this would be very expensive to calculate influence value for all out-of-sample augmentation data. We put the diversity filter ahead the influence value filter since the diversity filter would reduce the amount of data for IF value calculation.

4.4 Experimental Results

4.4.1 Two-stage Augmentation

The first two columns of Table 1 shows the experimental results that employing both in-sample augmentation data and out-of-sample augmentation data on stance detection and Chinese sentiment analysis tasks. First, significant improvements are observed along with most methods utilizing augmentation, compared with the baseline model BERT which does not take any augmentation data. Second, among the methods that use augmentation data, our method outperforms others in both tasks by substantial margins. Specifically, our methods surpass BERT-GradSim and Yang et al. (2020), which are the second best methods on each task, by 3.2 and 2.2 points respectively, showing that our method is better at utilizing different types of augmentation data, selecting samples with high effectiveness which leads to high performance. This is also confirmed by comparing the performance of BERT-mix and BERT-RandomAug, where BERT-RandomAug also samples in-sample and out-of-sample augmentation data separately. Given the same amount of augmentation data, Ours-noCL outperforms BERT-GradSim, which indicates that the Hessian matrix provides more information than the first-order gradient when it comes to approximating generalization errors. Finally, comparing the last two rows of the first two columns, the gain brought by IF is very large. Removing IF for SD results in a 11 points drop, and for CNSA 7 points drop, indicating approximating data effectiveness through IF estimation is crucial for the performance of the algorithm.

The experimental results on using out-of-sample or in-sample augmentation data separately in these two tasks are shown in the last four columns in Table 1. Performance of models with out-of-sample augmentation data only is generally worse than in-sample augmentation data only, mainly due to the combination of initial model being only trained with a small amount of gold data, a high amount of noise and the domain difference between the augmentation data and the development sets used for evaluation. However, our methods still perform the best compared to the baselines, confirming the flexibility of our proposed algorithm in dealing with different types of augmentation data. Similar performance drops are also observed when IF is ablated out.

4.4.2 In-sample Augmentation with GLUE

The experimental results on the in-sample augmentation over 6 GLUE benchmarks are shown in the Table 2. Our method outperforms other baseline models on 5 tasks. Particularly in the CoLA task, in which implementing data augmentation is difficult, all methods except our method show worse perfor-

	SD-IS-OOS	CNSA-IS-OOS	SD-IS	SD-OOS	CNSA-IS	CNSA-OOS
BERT	.555	.349	.554	.554	.347	.347
BERT-mix	.558	.355	.562	.546	.354	.324
BERT-RandomAug	.570	.361	.565	.560	.351	.356
BERT-GradSim	.635	.369	.591	.589	.361	.362
Yang et al. (2020)	.565	.401	.451	.469	.398	.395
Ours	**.667**	**.423**	**.614**	**.596**	**.417**	**.402**
Ours-noCL	.639	.372	.596	.595	.363	.387
Ours-noIF	.556	.353	.567	.590	.345	.341

Table 1: Experimental results on the development sets of stance detection and Chinese sentiment analysis task. The two-stage augmentation strategy helps our method achieve a large gain over that with only in-sample augmentation(IS) or out-of-sample augmentation(OOS), which is in support of the advantage of two-stage augmentation training. Our method is the best model with a large gap over other methods for both tasks.

mance than BERT without data augmentation.

The performance of Yang et al. (2020) is worse than others among all tasks except QQP, this is partially because the epoch-wise augmentation strategy is sensitive to the error of influence value estimation, which pushes the classification model away from the stage that has already been acquired with clean train data, and the negative impact would be accumulated in the progress of training. Another reason would be the self-supervised labeling makes the model be more confident to its incorrect prediction. Besides the CoLA task, BERT-RandomAug, Ours-noIF, and BERT-GradSim also achieve slightly worse results than BERT on the tasks of SST and QQP. One possible reason is that the BERT model has already achieved good results and these three methods cannot detect effective augmentation data to further improve performance, but the noise in the selected data degrades classification performance.

5 Analysis

Three ablation studies are conducted to explore the effect of training data size, hessian matrix update frequency and augmentation strategy on the effectiveness of the algorithm.

5.1 Sensitivity to training data size

In order to verify the sensitivity of the method to different training sizes, we down-sample the training set of the stance detection task to sub-set of size 200, 50 and 1. Given the same validation data and augmentation data source, our method's performance on SD-OOS is illustrated in Table 3. Our method performs better than the Bert-RandomAug baseline with a large margin in the scenarios of 50 training samples and 1 training sample. Our method shows competitive performance with only 1 training instance to the baseline model with 50 training samples, indicating the consistent performance of our method across various training sizes.

5.2 Inverse Hessian matrix update

The calculation of inverse hessian matrices is the major computation burden of our method. We conduct a study to evaluate how the update frequency of inverse hessian matrices influences the classification performance. Given the SD-OOS task, we tested three cases: no update in each training epoch where the calculation of the inverse hessian matrix happens in the beginning of each epoch, update per 5 training iterations and update every iteration. Table 4 shows the performance of our method in each case. The performance gap between update per iteration and update per epoch is fairly small. Therefore, we are able to calculate the inverse hessian matrix every epoch and use that matrix consistently in each training epoch, greatly reducing computation without losing much performance.

5.3 Epoch versus iteration-wise augmentation

We verify the performance of epoch-wise augmentation and iteration-wise augmentation on the out-of-sample augmentation for the stance detection task. Epoch-wise augmentation means $r|D_{\text{train}}|$ augmentation instances are selected at the beginning of each training epoch and feed the learning model with the selected augmentation data. Iteration-wise augmentation requires that the augmentation samples are selected after each training mini-batch, and the amount of selected augmentation samples equals to the division between augmentation budget and training iterations per epoch. Table 5 confirms the advantage of iteration-wise augmentation strategy, further confirming the dy-

	CoLA (MCC)	SST (ACC)	QQP (ACC)	MNLI-m (F1)	QNLI (ACC)	RTE (ACC)	Average
BERT	.355	.826	.693	.429	.728	.580	.602
BERT-mix	.197	.828	.687	.434	.731	.585	.577
BERT-RandomAug	.258	.829	.688	.439	.745	.586	.591
BERT-GradSim	.200	.825	.692	.431	.734	.584	.578
Yang et al. (2020)	.117	.759	.672	.394	.708	.540	.532
Ours	**.361**	**.831**	**.697**	.438	**.752**	**.591**	**.612**
Ours-noCL	.354	.829	.694	.445	.736	.586	.607
Ours-noIF	.156	.820	.638	**.449**	.738	.565	.561

Table 2: Experimental results on the dev sets of 6 GLUE benchmarks. Given in-sample augmentation data, our method outperforms or obtains similar performance with other models on all tasks except MNLI-m.

Training size	200	50	1
Ours	.596	.516	.424
Bert-RandomAug	.560	.412	.218

Table 3: Experimental results on dev sets of the stance detection task with different training size but the same out-of-sample augmentation data source. Our method outperforms the BERT-RandomAug baseline.

Update frequency	**0**	**per 5 steps**	**per iteration**
F1	.596	.607	.611

Table 4: Experimental results on the development sets of stance detection with different frequency of inverse hessian matrix update.

namic nature of the data effectiveness and the importance of model learning stages in utilization of augmentation data.

Augmentation strategy	epoch-wise	iteration-wise
F1	.532	.596

Table 5: Experimental results on the development sets of SD-OOS. Iteration-wise augmentation strategy is much better than the epoch-wise augmentation strategy.

5.4 Computational Complexity Analysis

The time complexity of our method is mainly determined by the stochastic estimation of $H_{\theta_t}^{-1}\nabla_\theta L(\mathrm{D}_{\text{valid}};\theta_t)$. The time complexity of stochastic estimation to the influence function is $O(ND + RKD)$ where N is the number of training samples and D is the dimension of model parameters. The stochastic estimation would run R times to achieve a stable inverse hessian matrix and $K(K < N)$ is the number of training samples used in each estimation iteration. The real time cost for the stance detection task is shown in A.2.

6 Conclusion

In this paper, we propose an augmentation data selection method to improve PLMs-based few-shot text classifications through dynamically choosing effective in-sample and out-of-sample augmentation data in accordance to the learning stage of classification models. The experimental results confirm the effectiveness of our method. Given these two augmentation sources, our method achieves large performance gain by using the two-stage augmentation training strategy together with the dynamic data selection algorithm.

7 Limitations

Despite the good performance of our proposed method, there are several limitations in our method. The influence value estimation is computationally expensive. Our method makes an augmentation decision on the basis of the optimal gradient descent direction, but the iterative training between augmentation data and training data would push the model from the optimal direction because of the high data variance in training set. The scope of in-sample augmentation data and out-of-sample augmentation data is strictly restricted, so it is worthy to explore mixing these two augmentation data sources in order to simultaneously accelerate convergence and improve generalizations.

8 Acknowledgement

This material is based in part upon work supported by the National Science Foundation under Grant IIS-2212174, IIS-1749940, Office of Naval Research N00014-20-1-2382, and National Institute on Aging (NIA) RF1AG072449.

References

Minmin Chen, Kilian Q Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. *Advances in neural information processing systems*, 24.

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. Advaug: Robust adversarial augmentation for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5961–5970.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

Yifan Fu, Xingquan Zhu, and Bin Li. 2013. A survey on instance selection for active learning. *Knowledge and information systems*, 35(2):249–283.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830.

Hongyu Guo. 2020. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4044–4051.

William W Hager. 1989. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239.

Junyuan Hong, Lingjuan Lyu, Jiayu Zhou, and Micheal Spranger. 2022. Outsourcing training without uploading data via efficient collaborative open-source sampling. In *NeurIPS*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*.

Amit Jindal, Narayanan Elavathur Ranganatha, Aniket Didolkar, Arijit Ghosh Chowdhury, Di Jin, Ramit Sawhney, and Rajiv Ratn Shah. 2020. Speechmix-augmenting deep sound recognition using hidden space interpolations. In *INTERSPEECH*, pages 861–865.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

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

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 31–41.

Guanyi Mou, Yichuan Li, and Kyumin Lee. 2021. Reducing and exploiting data augmentation noise through meta reweighting contrastive learning for text classification. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 876–887. IEEE.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in Neural Information Processing Systems*, 34:11054–11070.

Luis Perez and Jason Wang. 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Yanru Qu, Dinghan Shen, Yelong Shen, Sandra Sajeev, Jiawei Han, and Weizhu Chen. 2020. Coda: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. *arXiv preprint arXiv:2010.08670*.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33:596–608.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China national conference on Chinese computational linguistics*, pages 194–206. Springer.

Mengying Sun, Jing Xing, Bin Chen, and Jiayu Zhou. 2020. Robust collaborative learning with noisy labels. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1274–1279. IEEE.

Xinyi Wang, Hieu Pham, Paul Michel, Antonios Anastasopoulos, Jaime Carbonell, and Graham Neubig. 2020. Optimizing data usage via differentiable rewards. In *International Conference on Machine Learning*, pages 9983–9995. PMLR.

Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. 2021. Meta self-training for few-shot neural sequence labeling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1737–1747.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, Hong Kong, China. Association for Computational Linguistics.

Xiangpeng Wei, Heng Yu, Yue Hu, Rongxiang Weng, Weihua Luo, Jun Xie, and Rong Jin. 2022. Learning to generalize to more: Continuous semantic augmentation for neural machine translation. *arXiv preprint arXiv:2204.06812*.

Xiangpeng Wei, Heng Yu, Yue Hu, Rongxiang Weng, Luxi Xing, and Weihua Luo. 2020. Uncertainty-aware semantic augmentation for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2724–2735.

Sen Wu, Hongyang Zhang, Gregory Valiant, and Christopher Ré. 2020. On the generalization effects of linear transformations in data augmentation. In *International Conference on Machine Learning*, pages 10410–10420. PMLR.

Xing Wu, Chaochen Gao, Meng Lin, Liangjun Zang, and Songlin Hu. 2022. Text smoothing: Enhance various data augmentation methods on text classification tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 871–875.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33:6256–6268.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. G-daug: Generative data augmentation for commonsense reasoning. In *EMNLP (Findings)*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. 2021. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Rong Zhang and Alexander I Rudnicky. 2006. A new data selection principle for semi-supervised incremental learning. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 780–783. IEEE.

Jing Zhou, Yanan Zheng, Jie Tang, Jian Li, and Zhilin Yang. 2021. Flipda: Effective and robust data augmentation for few-shot learning.

Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. 2020. Learning data augmentation strategies for object detection. In *European conference on computer vision*, pages 566–583. Springer.

A Appendix

A.1 Hyperparameter Settings

Hyperparameters	Setting
Optimizer	AdamW
Adam β_1	0.9
Adam β_2	0.98
Adam ϵ	1e-3
Learning rate	5e-5
Maximum training epochs	25
Weight decay	0.01
Batch size	8

Table 6: The hyper-parameter settings for fine-tuning the base BERT model.

We froze the *embedding layers* to reduce computational complexity.

A.2 Hardware and running time

We run experiments on a workstation with a Titan RTX GPU of 24GB memory. Given the stance detection task, for each epoch with 25 training iterations and a mini-batch of 8, the average training time of each epoch is around 285 seconds on the condition that the Hessian-vector product is only computed in the beginning of each training epoch

A.3 Augmentation budget ratio

We take $r = 1$ for all in-sample augmentation as well as the out-of-sample augmentation for the stance detection task, and $r = 0.5$ for out-of-sample augmentation of the Chinese sentiment analysis task.

A.4 Curriculum learning strategy parameters

$\tau = 0.9$ and $\eta = 0.1$ for in-sample augmentation.
$\tau = 0.9$ and $\eta = 0$ for out-of-sample augmentation.

A.5 out-of-sample augmentation data Samples

Target: if the sample stream is not substantially modified by the analyser, it can be returned to the process **Claim:** the sample stream is not returned; for example, if any reagents have been added for the analysis. **Label:** Against
Target: imports of high-tech products far exceed exports **Claim:** r & d intensity is defined as r & d expenditure divided by net sales. **Label:** None
Target: his acts are designed to maximize publicity **Claim:** he means to fuse his name forever to a place, a date, an event. **Label:** Favor

Table 7: out-of-sample augmentation samples for the stance detection task.

Chinese Sentiment Analysis	**Data:** 什么时候能看??? **Translation:** When can we see that??? **Label:** Angry **Data:** 2011过去了，2012要努力... **Translation:** 2011 is over, work hard on 2012... **Label:** Neutral **Data:** 快乐，谢谢，亲。 **Translation:** Nice, thanks, Dear. **Label:** Happy **Data:** 明早又要进基地了，这是真的吗？真的吗？真的吗? **Translation:** We need to return to the (practice) base, is this true? Is this true? Is this true? **Label:** Surprised **Data:** 终于把心理学粗略抄完了。。好像憔悴了十岁一样。。 **Translation:** Transcribed the psychology materials finally.. I am more like becoming ten years older.. **Label:** Sad **Data:** 不想上学。 **Translation:** Do not want to go to school. **Label:** Fearful

Table 8: out-of-sample augmentation samples for the Chinese sentiment analysis task.