Scale-invariant-Fine-Tuning (SiFT) for Improved Generalization in Classification

Luheng (Wes) Wang [*]
New York University
lw2534@nyu.edu

Yilun Kuang [†]
New York University
yk2516@nyu.edu

Yash Bharti[‡]
New York University
yb1025@nyu.edu

Abstract

In natural language processing, we are concerned about domain generalization of our model when it comes to real life scenarios. In addition, researchers find that adversarial training, while improves robustness, often restricts model's ability for generalization. The current state-of-the-art adversarial training algorithm SMoothness-inducing Adversarial Regularization and BRegman pRoximal poinT opTimization (SMART, (Jiang et al., 2020)) algorithm resolves this conflict. We suggest a variation of SMART, the Scale-invariant-Fine-Tuning (SiFT) which is inspired by the brief description in the DeBERTa (He et al., 2021) paper. Our implementation of SiFT shows slight increase in in-domain testing results. More importantly, our classifier's performance on out-of-domain datasets is improved by approximately 10% comparing to SMART.

1 Introduction

Domain generalization is a capability of a model that allows it to be trained in one domain and perform well in another unseen domain. The concrete definition of domain varies case by case. For example, it is intuitive to believe that a generalized model which performs well on Twitter sentiment classification should also obtain good results on IMDB comment sentiment datasets.

In the SMART paper, the authors first use multi-task learning (MTL, (Liu et al., 2019a)) with SMART to train shared embeddings to regularize and prevent over-fitting. They have the regular fine-tuned model on each task as the baseline. Then, they test the MTL model with SMART on SNLI and SciTail, which are considered out-domain tasks. The results show consistent improvement over the baselines.

[*] First Author
[†] Co-Second Author
[‡] Co-Second Author

More specifically, SMART consists of two sections. First, it introduces smoothness-inducing adversarial training. It creates a adversarial loss $R_s(\theta)$ which is defined to be the symmetrical KL-Divergence of $1)$ output of the model with parameters θ and input x, and $2)$ output of the model with the same set of parameters θ but different input $\tilde{x}$, which is the perturbed embeddings. SMART combines this adversarial loss and the regular model loss. Then it optimizes this combined loss so that the output of the model does not change much when a small perturbation is injected, thus smoothness. Second, SMART incorporates Bregman Proximal Point Optimization to optimize the regularized loss in order to avoid aggressive update. It is done by using symmetrical KL-Divergence again to monitor the change in the outputs when the parameter θ is updated in each iteration.

Our research is driven by the question: if the out-of-domain datasets contain words which have very different embeddings comparing to the in-domain dataset, how do we improve the SMART algorithm's generalization stability. SiFT becomes a solution to this problem, as it contains an extra step of normalization which effectively eliminates the variance explained above. The adversarial training procedure is the same in SiFT and SMART, except that the embeddings are perturbed after they are normalized by layer in SiFT. The normalization procedure used is Layer Normalization (Ba et al., 2016). Layer normalization deals with each token's embedding vector, which should be in the same dimension. For instance, in BERT$_{base}$ model, layer normalization manipulates the vectors which have dimension of 768.

Our implementation of SiFT splits the regular BERT and DeBERTa models into embedding layer and the rest. We process the output of the embedding layer in the following three steps: normalization, perturbation, de-normalization. Then we feed it into the rest of the model. The logits we obtain from this setup is the further used to calculate the adversarial loss. Our experiment first assures that SiFT does not hurt regular in-domain testing. Then we compare models' performances on an out-of-domain dataset with SiFT and SMART. We have proved that models with SiFT consistently outperform models with SMART on an out-of-domain dataset by approximately 10%. This shows that by first normalizing the embeddings before perturbing them improves models' ability to generalize.

2 Background

In recent years, pretrained language models fine-tuned for downstream tasks like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and DeBERTa (He et al., 2021) have achieved SOTA performance in various benchmarks like GLUE (Wang et al., 2019). However, fine-tuning algorithms sometimes overfit the data and result in poor generalization performance (Jiang et al., 2020).

2.1 Random Perturbation Regularization

Classical regularization techniques augment the optimization objective with a penalization term with a hyperparameter (Miyato et al., 2018). From a Bayesian perspective, the addition of the regularization terms serve as a prior belief of the conditional output distribution $p(y|x)$ of the model (Bishop 2006; Miyato et al. 2018). For the model to have good generalization performance, the conditional output distribution $p(y|x)$ of the model should be smooth relative to the conditional input x (Miyato et al., 2018). In other words, the model should produce roughly the same output distribution $p(y|x)$ if the input data x is isotropically perturbed (Miyato et al., 2018).

Exploiting the idea of random perturbation uniformly in all directions, it has been demonstrated that training neural network models with random noise perturbation to the input x is equivalent to Tikhonov regularization (Bishop, 1995).

Subsequent studies by Szegedy et al. (2014) and Goodfellow et al. (2015) show that random perturbation in the adversarial direction leads to dramatic changes in the output distribution. Algorithms like adversarial training in supervised settings are proposed to add perturbations to the most anisotropic direction in the optimization objective (Miyato et al., 2018).

2.2 Adversarial Training

It is suggested that adversarial examples consistently leads to misclassification in a varieties of machien learning and neural network models (Goodfellow et al., 2015). The adversarial training is then proposed with the optimization formulation as follows

$$\mathcal{L}_{\text{adv}}(x_l, \theta) := l_s[q(y|x_l), p(y|x_l + r_{\text{adv}}, \theta)] \quad (1)$$

where $l_s(q,p)$ is a function that measures the difference between two distribution $q(\cdot)$ and $p(\cdot)$, $q(y|x_l)$ is the true output distribution in the supervised setting, $p(y|x_l)$ is the output distribution of the model, and $\mathcal{D}_l = \{(x_l^{(n)}, y_l^{(n)})\}_{n=1}^{N_l}$ is a labeled dataset (Miyato et al., 2018). r_{adv} is the noise in the adversarial direction that maximize the loss l_s between two distributions. Notice that under L_∞ norm, $r_{\text{adv}} \approx \epsilon \,\text{sign}(\nabla_{x_l}\mathcal{D}[q(y|x_l), p(y|x_l, \theta)])$, which is the original fast gradient sign method regularization technique proposed by (Goodfellow et al., 2015) for adversarial trianing.

By minimizing the loss $\mathcal{L}_{\text{adv}}$ with respect to the model output distribution $p(y|x_l + r_{\text{adv}}, \theta)$ that is adversarially perturbated, the output distributions of the model are forced to be smooth along the anisotropic direction. Adversarial training thereby regularizes the model to be robust against adversarial perturbations (Miyato et al., 2018).

In addition, adversarial training in NLP also improves model generalization (Miyato et al., 2017; Cheng et al., 2019). Some prominent adversarial training algorithms like FreeLB (Zhu et al., 2020) uses projected gradient descent (PGD) to update the noise perturbations that results in more invariance in the embedding space and hence better model generalization. CLAPS (Lee et al., 2021) is also an example of adversarial contrastive learning that use adversarial inputs as negative examples, which also leads to improved generalization.

2.3 Virtual Adversarial Training

To extend adversarial training to semi-supervised learning domain, consider

$$\mathcal{L}_{\text{vadv}}(x_l, \theta) := l_s[q(y|x_*), p(y|x_* + r_{\text{qadv}}, \theta)] \quad (2)$$

where x_* comes from either the labeled dataset $\mathcal{D}_l = \{(x_l^{(n)}, y_l^{(n)})\}_{n=1}^{N_l}$ and the unlabeled dataset $\mathcal{D}_{ul} = \{x_{ul}^{(m)}\}_{m=1}^{N_{ul}}$ (Miyato et al., 2018). Given that $q(y|x_{ul})$ is not accessible in the semi-supervised setting, we can approximate $q(y|x_{ul})$

by the current estimate $p(y|x_*, \hat{\theta})$ which is a "virtual" label (Miyato et al., 2018). Then the averaged $\mathcal{L}_{\text{vadv}}$ can be augmented to the full loss as a regularization term. By extending adversarial training to the semi-supervised domain, we can then regularize the model against adversarial attacks.

2.4 SMART, ALUM, SiFT

Several virtual adversarial training algorithm like SMART (Jiang et al., 2020), ALUM (Liu et al., 2020), and SiFT (He et al., 2021) are proposed. SMART uses a similar adversarial regularization term with modification of symmetric KL-divergence as adversarial loss compared to the standard VAT adversarial loss. Bregman proximal point optimization is used to constrain aggressive parameter updates (Jiang et al., 2020).

ALUM builds on the SMART VAT training objective by replacing of the bregman proximal point optimization with a curriculum learning approach, which train the model using standard object first and then switch to virtual adversarial training (Liu et al., 2020). The curriculum learning approach leads to faster training time and spare the use of the Bregman proximal point method (Liu et al., 2020).

The Scale Invariant Fine-Tuning (SiFT) algorithm builds on SMART by applying perturbations to normalized embeddings (He et al., 2021). It is claimed that the extra normalization applied to the word embedding improve the generalization performance of the DeBERTa and DeBERTa$_{1.5B}$ model (He et al., 2021). In this paper, we explore the SiFT algorithm, which is one variant of the VAT algorithm proposed by (Miyato et al., 2018). By comparing SiFT with the standard fine-tuning baseline and SMART, we can see how SiFT differs from SMART in generalization improvement.

3 Method

As illustrated in the introduction, we make use of the output embeddings of the embedding layer in the model. We record the mean and standard deviation for each embedding during normalization step. After they are normalized, we move to the perturbation step, which follows the same structure as discussed in the SMART paper.

First, we initialize and add a vector of random noise with mean 0 and standard deviation 0.01 to the embeddings. Then we follow the same strategy as SMART to update the noise, which is to minimize the following $\mathcal{R}_s$.

$$\mathcal{R}_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{||\tilde{x_i}-x_i||\leq\epsilon} l_s(f(\tilde{x_i};\theta), f(x_i;\theta)), \tag{3}$$

where $f(\cdot)$ is the model, x_i is the word embedding, $\tilde{x}_i$ is the normalized perturbed word embeddings, θ is the parameter, and l_s is the symmetrized KL divergence loss. This measures the local Lipschitz continuity, or in other words induces smoothness.

Finally, we use the mean and standard deviation we stored to de-normalize the embeddings. This is to maintain consistency in later steps when we compare the adversarial logits with the regular logits which are obtained from non-normalized embeddings. Then we feed these de-normalized embeddings into the rest of the models, for which the model outputs adversarial logits. Then we use the equation 3 again to calculate the adversarial loss, and take the sum of it and the regular training loss for parameter update.

4 Data

We use the following datasets: 1) Twitter Hate Speech dataset[1], 2) UCI Sentiment dataset [2], 3) CoLA dataset (Warstadt et al., 2019). The usage will be detailed in Section 5.

5 Experiment Design and Baselines

We incorporated both DeBERTa and BERT. We test on DeBERTa because SiFT was originally proposed by the DeBERTa authors. We also test on BERT for generalization purpose.

First, we fine-tune BERT and DeBERTa on CoLA and with resulting Matthews Correlation Coefficient of 0.52 and 0.54 respectively. We record these results to assure that our SiFT implementation does not hurt in-domain training and testing. This is shown by comparing to the models' performances on CoLA after SiFT is implemented. If the results do not vary much, then we conclude SiFT does not affect in-domain training and testing.

Second, we define our baseline to be the following so as to show domain generalization. We fine-tune BERT and DeBERTa on Twitter Hate Speech dataset. In addition, after fine-tuning the two models on Twitter Hate Speech data set, we do another

[1]https://www.kaggle.com/arkhoshghalb-sentiment-analysis-hatred-speech

[2]http://archive.ics.uci.edu/ml/datasets.php

Model + Algorithm	CoLA	UCI 10%
Plain BERT	0.52	0.55
BERT w SMART	0.51	0.70
BERT w SiFT	0.53	0.80

Table 1: The middle column shows the performance of each model + corresponding algorithm (if any) on CoLA dataset. The rightmost column shows the performance of each model + corresponding algorithm (if any) on the remaining 90% of the UCI dataset, after it is trained on the entire Twitter dataset as well as 10% of the UCI dataset.

Model + Algorithm	CoLA	UCI 10%
Plain DeBERTa	0.54	0.56
DeBERTa w SMART	0.56	0.80
DeBERTa w SiFT	0.56	0.91

Table 2: Table structure is the same as Table 1, except this table is for DeBERTa.

fine-tuning on UCI sentiment dataset, but with only 10% of the data. This is a setup similar to that of the SMART paper. This shows quantitatively the adaption of the models on out-of-domain tasks. Finally, we test on the remaining UCI dataset to obtain a testing performance.

We implement both SiFT and SMART on both BERT and DeBERTa, and follow the strategy described above. We prove that the out-of-domain performance is improved by SiFT comparing to SMART.

6 Testing Results & Analysis

We do multiple runs and record the average performance, which is shown in Table 1 and 2. First, the performances of both models on CoLA with SiFT implemented remain close to the performances of the plain models. We conclude that SiFT is not affecting the model's in-domain performance.

Then we inspect the performance on UCI dataset to analyze the out-domain performance. The numbers under "UCI 10% " in Table 1 and 2 refer to the testing performances on the remaining 90% data from UCI dataset after the model is fine-tuned on the Twitter dataset and 10% of the UCI dataset. Note that the models with the SIFT algorithm outperform the models with the SMART algorithm and the plain models. For BERT, SiFT raises the performance from 0.70 of SMART to 0.80. For DeBERTa, it raises the performance from 0.80 of SMART to 0.91. Since the difference is significant and the results are generated upon multiple runs, we can safely conclude that SiFT has outperformed SMART in classification

7 Conclusion

With the results we obtained from our experiments, we can safely conclude the significance of SiFT, that it improves models' capability to generalize. Models trained with SiFT algorithm will perform better on domains they are not familiar with. However, there are limitations to our experiments. Our experiments were on three specific datasets. Our improvement in results can be confounded by the underlying distribution of these datasets. In addition, we have only tested the algorithm on two specific types of classification, while there are more to be explored, such as semantic entailment recognition. We leave the work of testing on more domain of data to the future scholars.

8 Ethical Considerations

The SiFT algorithm can be used in models other than BERT and DeBERTa, such as RoBERTa and ALBERT. In addition, SiFT can be incorporated into other types of tasks beyond classification, such as ranking, text generation, and even recommendation system with further modification. However, the usage of SiFT might lead to both benefits and risks. On the bright side, data leakage should no longer be necessary when models using SiFT achieves better generalization performance that avoids the necessity for companies to acquire more data illegally. Big tech companies can train the model on their own data generated by user inputs and then used the model elsewhere publicly. In addition, SiFT helps improve the stability and overall performance of a model on out-of-domain data. Thus, SiFT might effectively restrain the possibilities for data privacy breaching for some vicious institutions.

On the other hand, SiFT can enhance institution's ability to obtain information in areas they previously were blocked from. For instance, consider a domain of data which this institution originally had very limited access to. Now with SiFT the institution can fine-tune their model with very little data and achieve a decent functionality on that domain. In real life scenarios such as e-commerce platforms, SiFT with its normalization step minimizes the gap between different domains of data, so that malicious users might be able to follow such strategy to get relatively accurate portraits of users from rival e-commerce platforms. Another potential negative impact of SiFT is that it might reduce researcher's interest and need for acquiring detailed data, since embeddings will be normalized and perturbed. The level of detail in data might be less important with SiFT implemented, which has unpredictable influence on data mining.

9 Collaboration Statement

1. Luheng (Wes) Wang:
Implementation of SiFT and SMART on BERT;
Implementation of SiFT and SMART on DeBERTa;
Experiment of SMART and SiFT on DeBERTa ;
Write-up of method and experiment

2. Yilun Kuang:
Implementation of SiFT and SMART on BERT;
Literature review for adversarial training.

3. Yash Bharti:
Experiment of SMART and SiFT on UCI with BERT;
Experiment of SMART and SiFT on CoLA with BERT;
Write-up of experiment results

10 Github

All code files for the current project can be found here:
https://github.com/wangluheng328/SiFT-Project

11 Acknowledgements

We thank Jason Phang and Will Huang for their help on our project.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Chris M. Bishop. 1995. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116.

Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pretrained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Seanie Lee, Dong Bok Lee, and Sung Ju Hwang. 2021. Contrastive learning with adversarial perturbations for conditional text generation.

Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. 2020. Adversarial training for large neural language models.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding.

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

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification.

Takeru Miyato, Shin ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: A regularization method for supervised and semi-supervised learning.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding.