PATS: Sensitivity-aware Noisy Learning for Pretrained Language Models

Yupeng Zhang[1], **Hongzhi Zhang**[2], **Sirui Wang**[2], **Wei Wu**[2] and **Zhoujun Li**[1]*
[1]Beihang University, Beijing, China [2]Meituan Inc., Beijing, China
{G0vi_qyx, lizj}@buaa.edu.cn
{zhanghongzhi03, wangsirui, wuwei30}@meituan.com

Abstract

A wide range of NLP tasks benefit from the fine-tuning of pretrained language models (PLMs). However, a number of redundant parameters which contribute less to the downstream task are observed in a directly fine-tuned model. We think the gap between pretraining and downstream tasks hinders the training of these redundant parameters, and results in a suboptimal performance of the overall model. In this paper, we present PATS (**P**erturbation **A**ccording **T**o **S**ensitivity), a noisy training mechanism which considers each parameter's importance in the downstream task to help fine-tune PLMs. The main idea of PATS is to add bigger noise to parameters with lower sensitivity and vice versa, in order to activate more parameters' contributions to downstream tasks without affecting the sensitive ones much. Extensive experiments conducted on different tasks of the GLUE benchmark show PATS can consistently empower the fine-tuning of different sizes of PLMs, and the parameters in the well-performing models always have more concentrated distributions of sensitivities, which experimentally proves the effectiveness of our method.

1 Introduction

With a huge number of model parameters and well designed training objectives, pretrained language models (PLMs) have brought a new era to NLP (Guu et al., 2020; Liu, 2019; Zhu et al., 2020b; Qiu et al., 2020). Fine-tuning PLMs such as BERT (Devlin et al., 2019) has become a basic and effective way in many downstream tasks (Wadden et al., 2019; Sun et al., 2019; Howard and Ruder, 2018).

However, recent study has shown that aggressive fine-tuning can induce an unstable and suboptimal performance of the models especially with insufficient data (Dodge et al., 2020; Raffel et al., 2019), which attracts some researchers to figure out the culprits and explore effective methods to solve them (Peters et al., 2019; Houlsby et al., 2019; Mosbach et al., 2020). For example, there are some regularization methods like RecAdam (Chen et al., 2020) and Mixout (Lee et al., 2020), and adversarial training techniques like SMART (Jiang et al., 2020) and FreeLB (Zhu et al., 2020a) to alleviate the overfitting of data in downstream tasks; Beyond that, Wu et al. (2022) proposed NoisyTune with the argument that in addition to the overfitting of the limited downstream data, there could also exist overfitting in pretraining tasks, which could result in enormous gaps between pretraining and downstream task data. In order to overcome the gaps, NoisyTune simply adds some noise to parameters in the PLM before fine-tuning. Besides, it has also been demonstrated that the existence of a large number of redundant parameters could also be a factor in the suboptimal performances of aggressively fine-tuned PLMs (Fan et al., 2019; Sanh et al., 2020; Dalvi et al., 2020). Considering the redundant parameters in a model are not insufficiently trained, Liang et al. (2022) proposed a learning rate scheduler named SAGE in which larger learning rates are assigned to these parameters of low sensitivity (a measure of parameter's importance to downstream tasks).

There could be some connection between the gaps caused by overfitting of pretraining tasks and the redundancy of parameters. We consider it could be the gaps between pretraining and downstream tasks that hinder the training of these redundant parameters. SAGE enlarges the learning rates of insensitive parameters to help their training. However, with the sensitivity measurement considered, the insensitive parameters usually have smaller gradients, so enlarged learning rates may help them little to escape the sub-optimal areas compared to involving additional noise. One noisy training method to alleviate the gaps is NoisyTune, in which parameters of a matrix in a PLM are added with

*Corresponding author.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3680–3687
December 7-11, 2022

noise according to the standard deviation of the matrix before fine-tuning. Nevertheless, there are few explanations about why or whether the parameters in the same matrix should be perturbed with the same intensity. Considering different parameters have different contributions to the model, noise from a unified distribution may disturb knowledge of some sensitive parameters, resulting in a loss of performance. Besides, since each task needs to capture an appropriate textual pattern and the data of it usually comes from a special domain, different downstream tasks could have different kinds of gaps with those of the pretraining. So the noise added to overcome the gaps should also be related to the downstream task data.

In this paper, we propose a novel parameter-wise noisy fine-tuning method called PATS (**P**erturbation **A**ccording **T**o **S**ensitivity) to make full use of perturbation on parameters to handle the problems above. We focus on balancing the contributions of all parameters in the model by activating the insensitive ones to play better roles in downstream tasks. So the main idea of our method is adding different intensities of noise to parameters according to their sensitivity when fine-tuning PLMs, different from NoisyTune (Fig. 1 (b)) in which noise added to a matrix of parameters is from a unified distribution and unrelated to downstream task data. Specifically, during fine-tuning in PATS (Fig. 1 (c)), larger noise will be added to the parameters with lower sensitivity (such as the parameter shown in red), while sensitive parameters (such as the parameter shown in purple) will be barely perturbed.

Our contributions can be summarized as follows: 1) We propose a simple but effective method to help all parameters be trained sufficiently when fine-tuning PLMs in downstream tasks. 2) Among all the training methods with noise, PATS is the first sensitivity-aware one which perturbs models with noise of different distributions according to parameters' sensitivity, to the best of our knowledge. 3) Extensive experiments on the GLUE benchmark show PATS makes a difference in boosting the performance of PLMs in downstream NLP tasks.

2 Approach

In this section, we present our PATS for PLMs fine-tuning. Previous matrix-wise noisy methods perturb a PLM by adding noise from a uniform distribution to a matrix of parameters. Different

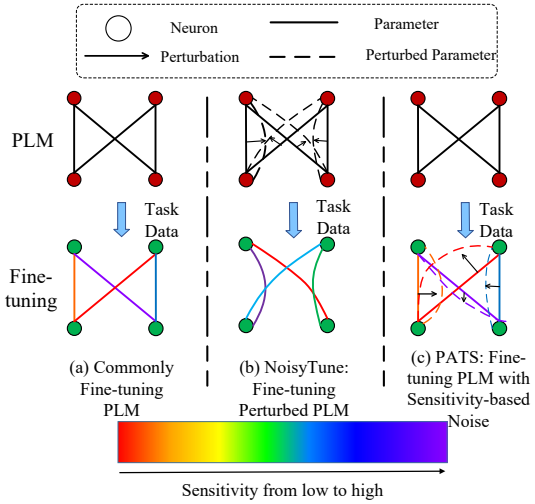

Figure 1: Different schemata of fine-tuning PLMs. In NoisyTune, a matrix of parameters in a PLM are perturbed with the same intensity before fine-tuning; In PATS, parameters with lower sensitivity (the "red" parameter) to downstream data are added with larger noise, and vice versa (like the "purple" parameter).

from them, in PATS, each parameter even from the same matrix will be paid to different attention according to its sensitivity. It is also worth noting that in PATS, a PLM is not perturbed in advance like NoisyTune, instead the perturbation happens during training as the task data comes in. In the following sections, we will introduce the calculation of parameter sensitivity first and then present the noisy learning mechanism in detail.

2.1 Sensitivity Measurement

The sensitivity of a parameter is used to measure the change of the output or loss after setting it to zero (Molchanov et al., 2017, 2019; Ding et al., 2019; Xiao et al., 2019; Lee et al., 2019). To be specific, given a BERT-like pre-trained language model $\mathbf{M}$ with parameters $\mathbf{\Theta} = \{\theta_1, \theta_2, \cdots, \theta_n\} \in \mathbb{R}^n$, the sensitivity of the j-th parameter θ_j is written as s_j, which can be defined as:

$$\begin{aligned} s_j &= |\mathcal{L}(\mathbf{\Theta}) - \mathcal{L}(\theta_1, \cdots, \theta_{j-1}, 0, \theta_{j+1}, \cdots, \theta_n)| \\ &\approx |\theta_j \nabla_{\theta_j} \mathcal{L}(\mathbf{\Theta})|, \end{aligned} \tag{1}$$

where $\mathcal{L}$ is a loss function and we use the first-degree Taylor polynomial to approximate s_j ignoring the higher order remainder to accelerate the calculation of it.

In order to avoid huge oscillation of s_j caused by an abnormal batch of data, we adopt the expo-

nential moving average of s_j used in many other models and optimizers (Liang et al., 2022; Klinker, 2010; Zhuang et al., 2022; Shahidi et al., 2020) as the real sensitivity indicator, which can be expressed by the following equation:

$$\bar{s_j} = \beta \bar{s_j}^* + (1-\beta)s_j, \beta \in (0,1), \quad (2)$$

where $\bar{s_j}$ and $\bar{s_j}^*$ are the exponential moving average of s_j in the current and previous iteration. β is a hyper-parameter used to adjust the importance of s_j calculated by the current batch of data.

2.2 Training with Noise

Algorithm 1 PATS for Adamax(max(·) returns a matrix with the maximum values of each element of the input matrices or vectors; sum(·) returns a scalar equal to the sum of all values of a matrix or vector; $\mathbf{I}$ denotes an all-ones matrix; $\odot$ denotes Hadamard product and $\oslash$ denotes Hadamard division)

Input: Step size α; Model parameters $\mathbf{\Theta} \in \mathbb{R}^n$; Number of training iterations T; Number of parameters in the current matrix N; Exponential decay rates $\beta, \beta_1, \beta_2 \in [0,1)$; Basic noise λ; Minimum effective sensitivity indicator γ; A small number that prevents an error of dividing by zero $\epsilon \in (0,1)$; Data $\mathcal{D}$; Loss function $\mathcal{L}(\cdot)$.

Initialize $\widetilde{\mathbf{S}}^{(0)} \leftarrow \mathbf{0} \in \mathbb{R}^N$.
Initialize $\mathbf{M}^{(0)} \leftarrow \mathbf{0} \in \mathbb{R}^N$.
Initialize $\mathbf{U}^{(0)} \leftarrow \mathbf{0} \in \mathbb{R}^N$.
for $t \leftarrow 1$ to T **do**
  $\mathbf{d}^{(t)} \overset{sample}{\longleftarrow} \mathcal{D}$.
  $\mathbf{G}^{(t)} \leftarrow \nabla_{\mathbf{\Theta}^{(t)}} \mathcal{L}(\mathbf{d}^{(t)}, \mathbf{\Theta}^{(t)})$.
  $\mathbf{S}^{(t)} \leftarrow \mathbf{\Theta}^{(t)} \odot \mathbf{G}^{(t)}$.
  $\mathbf{M}^{(t)} \leftarrow \beta_1 \mathbf{M}^{(t-1)} + (1-\beta_1)\mathbf{G}^{(t)}$.
  $\mathbf{U}^{(t)} \leftarrow \max(\beta_2 \mathbf{U}^{(t-1)}, |\mathbf{G}^{(t)}|)$.
  $\widetilde{\mathbf{S}}^{(t)} \leftarrow \beta \widetilde{\mathbf{S}}^{(t-1)} + (1-\beta)\mathbf{S}^{(t)}$.
  $\mathbf{R} \leftarrow \lambda \max(\text{sum}(\widetilde{\mathbf{S}}^{(t)})\mathbf{I} \oslash (N\widetilde{\mathbf{S}}^{(t)} + \epsilon \mathbf{I}) - \gamma \mathbf{I}, \mathbf{0})$.
  $\mathbf{Q} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$.
  $\mathbf{Z} \sim \mathcal{B}(N, p)$.
  $\mathbf{\Theta}^{(t+1)} \leftarrow \mathbf{\Theta}^{(t)} - (\alpha/(1-\beta_1^t))\mathbf{M}^{(t)} \oslash \mathbf{U}^{(t)} + \mathbf{Q} \odot \mathbf{Z}$.
  $t \leftarrow t+1$
end for

Our goal is to mainly activate the contributions of less sensitive parameters by perturbing them with bigger noise and leave parameters with larger sensitivity less affected at the same time. In our framework, we use a hyper-parameter λ as initial noise and the degree of perturbation to different parameters will be scaled up and down based on it according to their sensitivity. The intensity of perturbation can be formulated by the following equations:

$$\bar{s} = \frac{1}{N}\sum_{i=1}^{N} \bar{s_i} \quad (3)$$

$$r_j = \lambda \cdot \max(\frac{\bar{s}}{\bar{s_j} + \epsilon} - \gamma, 0), 0 < \epsilon \ll 1 \quad (4)$$

In Eq. 3, $\bar{s}$ is the average sensitivity of the matrix containing θ_j with N parameters. r_j in Eq. 4 means the intensity of the noise to be added on a parameter θ_j, which is scaled on λ by the division of $\bar{s}$ and $\bar{s_j}$. ϵ is a small number used to prevent zero denominator. Since r_j and $\bar{s_j}$ are inversely correlated, the intensity of noise added to every parameter with a lower sensitivity than the average will be larger than λ, and vice versa as we expect. As for the reason why $\bar{s}$ is restricted to the current matrix, as we found, the value distributions of different matrix parameters are sometimes very different. For example, values of parameters in matrix $\mathbf{A}$ are significant higher than those in matrix $\mathbf{B}$. And with the sensitivity measurement considered, sensitive parameters are usually themselves large on value. So if $\bar{s}$ is calculated based on all parameters of the model, some matrices of parameters with low values and sensitivity may be perturbed fiercely to some values very far from their original ones, which has unstable performances on experiments. To further reduce the perturbation on sensitive parameters and let them keep regular gradient-driven update, we use a margin constant γ in Eq. 4 to zero-out the noise added on the parameters that are highly sensitive.

For each parameter θ_j, the noise q_j that may finally be added to it is independently randomly sampled from a Gaussian distribution with the mean of zero and the standard deviation of σ_j as $q_j \sim N(0, \sigma_j^2)$, where $\sigma_j = \sqrt{r_j}$. So in an iteration, we update each parameter by:

$$\widetilde{\theta_j} = \theta_j - \eta \cdot \nabla_{\theta_j} \mathcal{L}(\mathbf{\Theta}) + q_j \cdot z, \quad (5)$$

where η is learning rate and $z \sim B(1,p)$ is a random value sampled from Bernoulli distribution which outputs 1 with probability p and 0 with probability $1-p$. Algorithm 1 shows the PATS algorithm for Adamax (Kingma and Ba, 2014) optimizer.

Model	CoLA	MRPC	RTE	STS-B	QQP	QNLI	MNLI	SST	Avg
	Mcc	F1	Acc	Pcc	F1	Acc	Acc	Acc	Score
$\text{BERT}_{\text{base}}$	58.94	90.19	68.03	89.28	88.53	91.96	84.63	92.77	83.18
$\text{BERT}_{\text{base}}$ + SAGE	59.45	90.53	71.78	89.81	88.61	91.87	84.59	**93.06**	83.65
$\text{BERT}_{\text{base}}$ + NoisyTune	60.01	90.34	69.71	89.81	88.58	91.82	84.64	92.85	83.47
$\text{BERT}_{\text{base}}$ + PATS	**60.67**	**91.05**	**72.08**	**89.86**	**88.64**	**92.02**	**84.80**	92.89	**84.00**
$\text{RoBERTa}_{\text{large}}$	66.67	91.89	85.44	91.98	89.26	94.45	90.25	96.10	88.25
$\text{RoBERTa}_{\text{large}}$ + SAGE	67.36	93.27	85.56	92.05	89.27	94.54	90.25	96.25	88.57
$\text{RoBERTa}_{\text{large}}$ + NoisyTune	67.47	93.26	85.52	92.00	89.36	94.53	90.04	96.12	88.56
$\text{RoBERTa}_{\text{large}}$ + PATS	**68.62**	**93.52**	**86.29**	**92.23**	**89.40**	**94.64**	**90.44**	**96.30**	**88.90**

Table 1: Results of models on GLUE dev set.

3 Experiments

3.1 Datasets and Baselines

We conduct extensive experiments on the eight tasks of the GLUE benchmark (Wang et al., 2018) and adopt the publicly available BERT-base (Devlin et al., 2019) and RoBERTa-large (Liu et al., 2019) models on every task individually. The following three baselines are selected for comparison: (1) **Standard PLM fine-tuning**, which fine-tunes PLMs directly; (2) **NoisyTune** (Wu et al., 2022), which is a noisy training method that adds matrix-wise noise before fine-tuning; (3) **SAGE** (Liang et al., 2022), which is an optimized learning rate schedule which adjusts the learning rate of every parameter according to its sensitivity.

3.2 Performance Evaluation

On each task, we repeat our experiments 5 times with different random seeds and report the average scores of every model, which are shown in Table 1.[1] According to the results, PATS optimized models consistently outperforms directly fine-tuned ones on different downstream tasks, especially on those with small datasets (CoLA & MRPC & RTE). Specifically, PATS improves by around 2 points on CoLA and RTE, and around 1 point on MRPC. In addition, as a parameter-wise method based on sensitivity, PATS experimentally outperforms the matrix-wise noisy method NoisyTune and the sensitivity-based learning rate scheduler SAGE on 7 out of the 8 tasks. The experimental results demonstrate the effectiveness of PATS.

[1]The results of the MNLI task are obtained by averaging the output accuracies of the models on the mnli-matched dataset and the mnli-mismatched dataset.

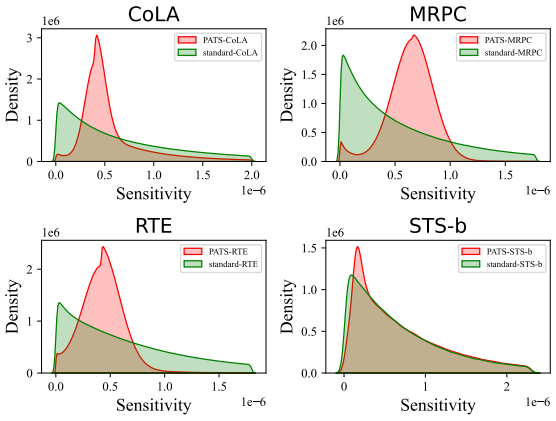

Figure 2: The sensitivity distributions of the parameters in different optimized models.

3.3 Empirical Analysis

In this section, we conduct additional analyses on sensitivity of parameters in the fine-tuned models.

Fig. 2 shows the sensitivity distribution of the model parameters fine-tuned in different ways. It is found that the sensitivity of the parameters in the PATS optimized models is more tightly clustered than that in the models fine-tuned in the common way. Besides, there remain fewer insensitive parameters in PATS optimized models than those in baseline models. And it is no longer a few high-sensitive parameters that dominate the models as what happens in normal fine-tuning, which indicates that perturbation helps parameters with low sensitivity gain more attention during training and lets the contribution of each parameter in the optimized models more balanced.

To further investigate the effect of PATS on small datasets, we also post the accuracies of the models fine-tuned on different proportions of training data sampled from the CoLA[2] dataset with and with-

[2]The phenomena observed on other tasks are similar.

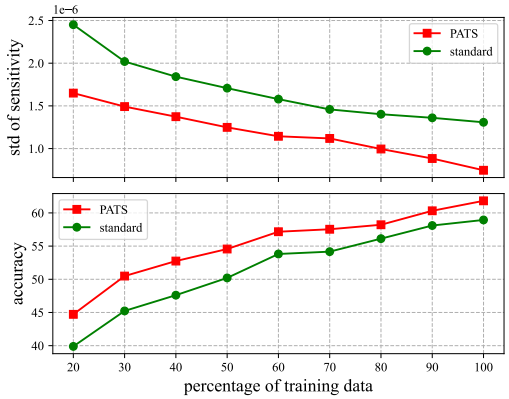

Figure 3: Performances of PATS on data of different sizes.

out PATS. Fig. 3 shows PATS optimized models consistently outperform directly fine-tuned ones on different sizes of datasets, demonstrating the generalizability of the approach. Moreover, we can also observe that as the size of training data increases, the performances of the models improve along with concomitant decreases in the standard deviations of sensitivity. This phenomenon further indicates that training with limited data will lose some of the performance capabilities of PLMs by leaving more undertrained or insensitive parameters, because small datasets is insufficient for PLMs to overcome the gap between pretraining and downstream tasks. The inverse correlation between accuracy and sensitivity concentration justifies our original intention of balancing the sensitivity of parameters. And the displayed performances experimentally demonstrate its availability.

4 Conclusion

We propose a novel noisy training method called PATS to optimize fine-tuning of PLMs. Since aggressive fine-tuning PLMs will leave a large number of insensitive parameters which contribute little to the overall model, PATS activates them and balance the contributions of all parameters in downstream tasks by adding noise to each parameter according to its sensitivity in the process of training. PATS is a simple mechanism without much computational and memory overhead compared to adversarial training which requires additional backwards passes. Extensive experiments on eight tasks of the GLUE benchmark show that PATS can consistently improve the performance of PLMs on downstream tasks with the sensitivity of the parameters more concentrated, which is especially pronounced on small datasets.

Limitations

PATS introduces four additional hyperparameters, which increases some work of users on hyperparameter tuning. For example, a too small λ could make few differences while an overlarge λ may result in unstable performances of models. Though we have summarized effective parameter configurations on the NLU tasks of the GLUE benchmark, it cannot guarantee that these settings are still applicable on other tasks such as neural machine translation. We will explore the connections between the hyperparameters in theory and narrow the search ranges of the hyperparameter group in future work.

References

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, and Hsiao-Wuen Hon. 2020. UniLMv2: Pseudo-masked language models for unified language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 642–652. PMLR.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*, volume 6, pages 6–4. Venice.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc Le, and Christopher D. Manning. 2020. Pre-training transformers as energy-based cloze models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 285–294, Online. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4908–4926, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaohan Ding, guiguang ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, and Ji Liu. 2019. Global sparse momentum sgd for pruning very deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2022.06305*.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. In *Advances in Neural Information Processing Systems*, volume 33, pages 9782–9793. Curran Associates, Inc.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Zhiqi Huang, Lu Hou, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2021. GhostBERT: Generate more features with cheap operations for BERT. 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 6512–6523, Online. Association for Computational Linguistics.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained 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.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Frank Klinker. 2010. Exponential moving average versus moving exponential average. *Mathematische Semesterberichte*, 58(1):97–107.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. 2019. SNIP: Single-Shot Network Pruning Based on Connectiono Sensitivity. In *International Conference on Learning Representations*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Chen Liang, Haoming Jiang, Simiao Zuo, Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Tuo Zhao. 2022. No parameters left behind: Sensitivity guided adaptive learning rate for training large transformer models. In *International Conference on Learning Representations*.

Yang Liu. 2019. Fine-tune BERT for extractive summarization. *CoRR*, abs/1903.10318.

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*.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11256–11264.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

XiPeng Qiu, TianXiang Sun, YiGe Xu, YunFan Shao, Ning Dai, and XuanJing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389. Curran Associates, Inc.

Hamidreza Shahidi, Ming Li, and Jimmy Lin. 2020. Two birds, one stone: A simple, unified model for text generation from structured and unstructured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3864–3870, Online. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. 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 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

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

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2022. NoisyTune: A little noise can help you finetune pretrained language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 680–685, Dublin, Ireland. Association for Computational Linguistics.

Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. 2019. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020a. Freelb: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020b. Incorporating bert into neural machine translation. *arXiv preprint arXiv:2002.06823*.

Weiming Zhuang, Yonggang Wen, and Shuai Zhang. 2022. Divergence-aware federated self-supervised learning. In *International Conference on Learning Representations*.

Model	COLA	MRPC	RTE	STS-B	QQP	QNLI	MNLI	SST
$\text{BERT}_{\text{base}}$	1e-4	1e-4	1e-4	2e-4	1e-4	2e-4	8e-5	8e-5
$\text{RoBERTa}_{\text{large}}$	3e-5	5e-5	5e-5	5e-5	1e-4	1e-5	3e-5	3e-5
$\text{BERT}_{\text{base}}$+PATS	1e-4	3e-4	3e-4	3e-4	2e-4	2e-4	1e-4	3e-4
$\text{RoBERTa}_{\text{large}}$+PATS	8e-5	8e-5	8e-5	5e-5	1e-4	3e-5	1e-5	1e-5

Table 2: Learning rate settings for PATS on the tasks of the GLUE benchmark.

Hyperparameters	Range
λ	{5e-7, 8e-7, 1e-6, 2e-6, 3e-6}
γ	{1e-3, 2e-3, 3e-3, 5e-3, 8e-3,2e-2}
β	{0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85}
learning rate	{1e-5, 3e-5, 5e-5, 7e-5, 8e-5, 1e-4, 2e-4, 3e-4, 5e-4}

Table 3: Searching ranges of hyperparameters in our experiments.

A Appendix

A.1 Datasets

The experiments are conducted on the GLUE benchmark, which contains several types of Natural Language Understanding (NLU) tasks such as linguistic acceptability (CoLA, Warstadt et al. 2019), text similarity (STS-B, Cer et al. 2017) and natural language inference (RTE & MNLI & QNLI, Dagan et al. 2005; Bar-Haim et al. 2006; Giampiccolo et al. 2007; Bentivogli et al. 2009; Williams et al. 2018; Bowman et al. 2015; Rajpurkar et al. 2016) tasks. Among the nine tasks, WNLI (Levesque et al., 2012) task is excluded in our experiments, on which BERT-like models have no obvious advantage over other mainstream baselines (Hou et al., 2020; Clark et al., 2020; Huang et al., 2021). Consistent with previous works (Bao et al., 2020; Wu et al., 2022), we evaluate results on the dev set of GLUE.

A.2 Training Details

For all the baseline models and our proposed PATS, we adopt a linear-decay learning rate schedule and choose Adamax (Kingma and Ba, 2014) which is the best-performing optimizer for baseline models on the GLUE benchmark to optimize the training. In PATS, we perturb the parameters of all the encoder layers except the Layer Normalization layers. In our training process, we set $\lambda = 2 \times 10^{-6}$, $\gamma = 0.002$, $\beta = 0.75$, $p = 0.2$ for all tasks. In addition, we adopt a linear warm-up learning rate schedule with 0.1 of total training iterations. The batch size of models is uniformly set to 32. We post the best performance models on each task after 10 epochs of training. The learning rates that yields the best generalization performance of models optimized by PATS and Standard PLM fine-tuning on each task are listed in Table 2. We present the searching range of hyperparameters in Table 3.

Our implementation is based on the MT-DNN code-base.[3]. And we use Nvidia V100 GPUs for all experiments.

A.3 Other Implemention Details

All datasets of the GLUE benchmark are downloaded from https:// gluebenchmark.com/tasks. For the baseline model SAGE, we use the code from the Github respository https://github.com/cliang1453/SAGE. The other baseline models are implemented by ourselves.

For the distribution of sensitivity shown in Fig. 2, we discard some outliers and only choose the parameters with sensitivity in the range of [5e-8, 1e-5] for visualization.

[3]https://github.com/namisan/mt-dnn