On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines

Marius Mosbach
Saarland Informatics Campus, Saarland University
mmosbach@lsv.uni-saarland.de

Maksym Andriushchenko
École polytechnique fédérale de Lausanne
maksym.andriushchenko@epfl.ch

Dietrich Klakow
Saarland Informatics Campus, Saarland University
dietrich.klakow@lsv.uni-saarland.de

Abstract

Fine-tuning pre-trained transformer-based language models such as BERT has become a common practice dominating leaderboards across various NLP benchmarks. Despite the strong empirical performance of fine-tuned models, fine-tuning is an unstable process: training the same model with multiple random seeds can result in a large variance of the task performance. Previous literature (Devlin et al., 2019; Lee et al., 2020; Dodge et al., 2020) identified two potential reasons for the observed instability: catastrophic forgetting and a small size of the fine-tuning datasets. In this paper, we show that both hypotheses fail to explain the fine-tuning instability. We analyze BERT, RoBERTa, and ALBERT, fine-tuned on three commonly used datasets from the GLUE benchmark and show that the observed instability is caused by optimization difficulties that lead to vanishing gradients. Additionally, we show that the remaining variance of the downstream task performance can be attributed to differences in generalization where fine-tuned models with the same training loss exhibit noticeably different test performance. Based on our analysis, we present a simple but strong baseline that makes fine-tuning BERT-based models significantly more stable than previously proposed approaches. Code to reproduce our results is available online: https://github.com/uds-lsv/bert-stable-fine-tuning.

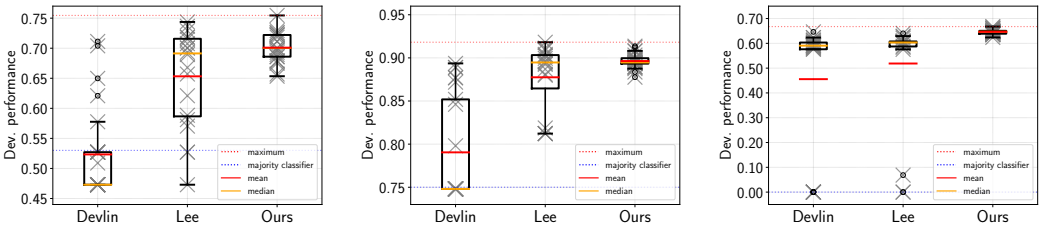
1 Introduction

Pre-trained transformer-based masked language models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and ALBERT (Lan et al., 2020) have had a dramatic impact on the NLP landscape in the recent year. The standard recipe of using such models typically involves training a pre-trained model for few epochs on a supervised downstream dataset. A process referred to as fine-tuning.

While fine-tuning has led to impressive empirical results, dominating a large variety of English NLP benchmarks such as GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a) it is still poorly understood. Not only have fine-tuned models been shown to pick up spurious patterns and biases present in the training data (Niven and Kao, 2019; McCoy et al., 2019), but also to exhibit a large training instability: fine-tuning a model multiple times on the same dataset, varying only the random seed, leads to a large standard deviation of the fine-tuning accuracy (Devlin et al., 2019; Dodge et al., 2020).

Few methods have been proposed to solve the observed instability (Phang et al., 2018; Lee et al., 2020), however without providing a sufficient understanding of why fine-tuning is prone to such

Preprint. Under review.



(a) RTE (b) MRPC (c) CoLA

Figure 1: Our proposed fine-tuning strategy leads to very stable results with very concentrated development set performance over 25 different random seeds across all three datasets on $\text{BERT}_{\text{LARGE}}$. In particular, we significantly outperform the recently proposed approach of Lee et al. (2020) in terms of fine-tuning stability.

failure. The goal of this work is to address this shortcoming. More specifically, we investigate the following question:

Why is fine-tuning prone to failures and how can we improve its stability?

We start by investigating two common hypotheses for fine-tuning instability: catastrophic forgetting and a small size of the fine-tuning datasets and demonstrate that both hypotheses fail to explain fine-tuning instability. We then investigate fine-tuning failures on three datasets from the popular GLUE benchmark and show that the observed fine-tuning instability can be decomposed into two unique aspects: (1) optimization difficulties early in training, characterized by vanishing gradients, and (2) differences in generalization late in training, characterized by a large variation of development set accuracy for runs with almost equivalent training performance.

Based on our analysis, we present a simple but strong baseline for fine-tuning pre-trained language models that significantly improves the fine-tuning stability compared to previous works (Fig. 1). Moreover, we show that our findings apply not only to the widely used BERT model but also to more recent models such as RoBERTa and ALBERT.

2 Related work

The fine-tuning instability of BERT has been pointed out in various studies. Devlin et al. (2019) report instabilities when fine-tuning $\text{BERT}_{\text{LARGE}}$ on small datasets and resort to performing multiple restarts of fine-tuning and selecting the model that performs best on the development set. Recently, Dodge et al. (2020) performed a large-scale empirical investigation of the fine-tuning instability of BERT. They found dramatic variations in fine-tuning accuracy across multiple restarts and argue how it might be related to the choice of random seed and the dataset size.

Few approaches have been proposed to directly address the observed fine-tuning instability. Phang et al. (2018) study intermediate task training (STILTS) before fine-tuning with the goal of improving performance on the GLUE benchmark. They also find that their proposed method leads to improved fine-tuning stability. However, due to the intermediate task training, their work is not directly comparable to ours. Lee et al. (2020) propose a new regularization technique termed Mixout. The authors show that Mixout improves stability during fine-tuning which they attribute to the prevention of catastrophic forgetting.

Another line of work investigates optimization difficulties when pre-training transformer-based language models (Xiong et al., 2020; Liu et al., 2020). Similar to our work, they highlight the importance of warmup for optimization. Both works focus on pre-training and we hence view them as orthogonal to our work.

3 Background

We first present the datasets used for fine-tuning and briefly recall the overall experimental protocol of fine-tuning pre-trained masked language models such as BERT.

3.1 Datasets

We study three datasets from the GLUE benchmark (Wang et al., 2019b): CoLA, MRPC, and RTE. Statistics for each of the three datasets can be found in Section 7.1 in the appendix.

CoLA. The Corpus of Linguistic Acceptability (Warstadt et al., 2018) is a sentence-level classification task containing sentences labeled as either grammatical or ungrammatical. Fine-tuning on CoLA was observed to be particularly stable in previous work (Phang et al., 2018; Dodge et al., 2020; Lee et al., 2020). Performance on CoLA is reported in Matthew's correlation coefficient (MCC).

MRPC. The Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) is a sentence-pair classification tasks. Given two sentences, a model has to judge whether the sentences paraphrases of each other. Performance on MRPC is measured using the average of accuracy and F_1 score.

RTE. The Recognizing Textual Entailment dataset is a collection of sentence-pairs collected from a series of textual entailment challenges (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009). RTE is the second smallest dataset in the GLUE benchmark and fine-tuning on RTE was observed to be particularly unstable (Phang et al., 2018; Dodge et al., 2020; Lee et al., 2020). Accuracy is used to measure performance on RTE.

3.2 Fine-tuning

Unless mentioned otherwise, we follow the default fine-tuning strategy recommended by Devlin et al. (2019): we fine-tune uncased $\text{BERT}_{\text{LARGE}}$ (henceforth BERT) using a batch size of 16 and a learning rate of $2\text{e}{-5}$. The learning rate is linearly increased from 0 to $2\text{e}{-5}$ for the first 10% of iterations—which is known as *warmup*—and linearly decreased to 0 afterwards. We apply dropout with probability $p = 0.1$ and weight decay with $\lambda = 0.01$. We train for 3 epochs on all datasets and use global gradient clipping. Following Devlin et al. (2019), we use the AdamW optimizer (Loshchilov and Hutter, 2019) *without* bias correction.

We fine-tune $\text{RoBERTa}_{\text{LARGE}}$ (Liu et al., 2019) and $\text{ALBERT}_{\text{LARGE-V2}}$ (Lan et al., 2020) using the same strategy. We note that compared to BERT, both RoBERTa and ALBERT have slightly different hyperparamters which we changed accordingly. RoBERTa uses weight decay with $\lambda = 0.1$ and no gradient clipping. ALBERT does not use dropout. A detailed list of all default hyper-parameters for all models can be found in Section 7.2 of the appendix.

Failed runs. Following Dodge et al. (2020), we refer to a fine-tuning run as a *failed run* if its accuracy (either at the end of training or after early stopping) is less or equal to that of a majority class classifier on the respective dataset. Majority baselines for all tasks are found in the appendix.

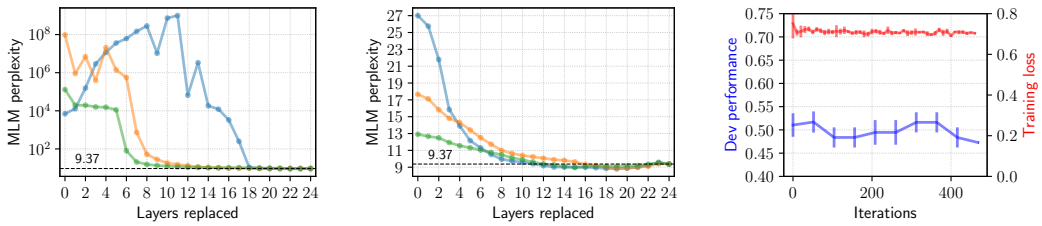
Implementation. For our experiments, all models were trained on single NVIDIA V100 GPUs. Our implementation is based on HuggingFace's transformers library (Wolf et al., 2019) and the code to reproduce our results is available at `https://github.com/uds-lsv/bert-stable-fine-tuning`.

4 Investigating previous explanations of fine-tuning instability

Previous works on fine-tuning predominantly state two hypotheses for what causes fine-tuning instability: *catastrophic forgetting* and *small training data size* of the downstream tasks. Despite the ubiquity of these hypotheses (Devlin et al., 2019; Phang et al., 2018; Dodge et al., 2020; Lee et al., 2020; Zhu et al., 2020), we argue that that none of them can fully explain the fine-tuning instability.

4.1 Does catastrophic forgetting cause fine-tuning instability?

Catastrophic forgetting (McCloskey and Cohen, 1989; Kirkpatrick et al., 2017) refers to the phenomenon when a neural network is sequentially trained to perform two different tasks, and it loses its ability to perform the first task after being trained on the second. More specifically, in our setup it means that after fine-tuning a pre-trained model, it can no longer perform the original masked language modeling task used for pre-training. This can be measured in terms of the perplexity on the original training data. Although the language modeling performance of a pre-trained model correlates with its fine-tuning accuracy (Liu et al., 2019; Lan et al., 2020), there is no clear motivation for why preserving the original masked language modeling performance after fine-tuning is important.



(a) Perplexity of failed models (b) Perplexity of successful models (c) Training of failed models

Figure 2: Language modeling perplexity for three failed (a) and successful (b) fine-tuning runs of BERT on RTE where we replace the weights of the top-k layers with their pre-trained values. We can observe that it is often sufficient to reset around 10 layers out of 24 to recover back the language modeling abilities of the pre-trained model. (c) shows the average training loss and development accuracy ($\pm$1std) for 5 failed fine-tuning runs on RTE. Failed fine-tuning runs lead to a trivial training loss suggesting an optimization problem.

Lee et al. (2020) directly motivate their approach as to prevent catastrophic forgetting, thus it is important to understand how exactly it occurs in fine-tuning and how it relates to the observed fine-tuning instability. To understand this, we perform the following experiment. We fine-tune BERT on RTE, following the default strategy by Devlin et al. (2019). We select three successful and three failed fine-tuning runs and evaluate their masked language modeling perplexity on the test set of the WikiText-2 language modeling benchmark (Merity et al., 2016).[1] We sequentially substitute the top-k layers of the network varying k from 0 (i.e. all layers are from the fine-tuned model) to 24 (i.e. all layers are from the pre-trained model).

We show the results in Fig. 2 (a) and (b). We can observe that although catastrophic forgetting occurs for the failed models (Fig. 2a) — perplexity on WikiText-2 is indeed degraded for $k = 0$ — the phenomenon is much more nuanced. Namely, catastrophic forgetting affects only the top layers of the network — in our experiments often around 10 out of 24 layers, and the same is however also true for the successfully fine-tuned models, except for a much smaller increase in perplexity.

Another important aspect of our experiment is that catastrophic forgetting typically requires that the model at least successfully learns how to perform the new task. However, this is not the case for the failed fine-tuning runs. Not only is the development accuracy equal to that of the majority classifier, but also the training loss on the fine-tuning task (here RTE) is trivial, i.e. close to $-\ln(1/2)$ (see Fig. 2 (c)). This suggests that the observed fine-tuning failure is rather an optimization problem *causing* catastrophic forgetting in the top layers of the pre-trained model. We will show later that the optimization aspect is actually sufficient to explain most of the variance in the fine-tuning performance.

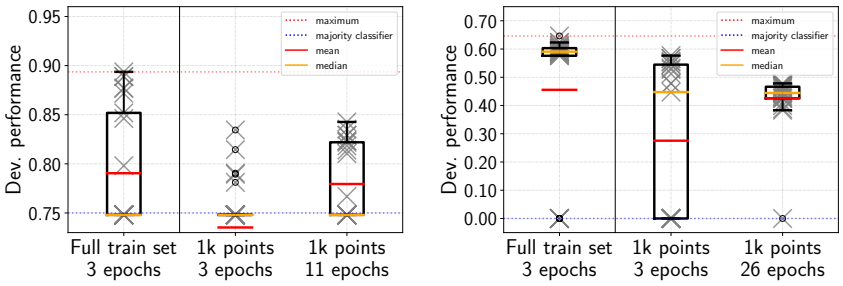
4.2 Do small training datasets cause fine-tuning instability?

Having a small training dataset is by far the most commonly stated hypothesis for fine-tuning instability. Multiple recent works (Devlin et al., 2019; Phang et al., 2018; Lee et al., 2020; Zhu et al., 2020; Dodge et al., 2020; Pruksachatkun et al., 2020) that have observed BERT fine-tuning to be unstable explicitly attribute this finding to the small number of training examples.

To test if having a small training dataset actually causes instability we perform the following experiment:[2] we randomly sample 1,000 training samples from the CoLA and MRPC training datasets and fine-tune BERT for 25 random seeds on each dataset. We compare two different setting: first, training for 3 epochs on the reduced training dataset and second, training for the same number of *iterations* as on the full training dataset. We show the results in Fig. 3. Note that training on less data does indeed affect the fraction of failed runs. However, when we train for as many iterations as on the full training dataset, we obtain back the original level of fine-tuning stability. Further, as expected, we observe that training on less samples affects the generalization of the model, leading to a worse validation performance on both tasks.

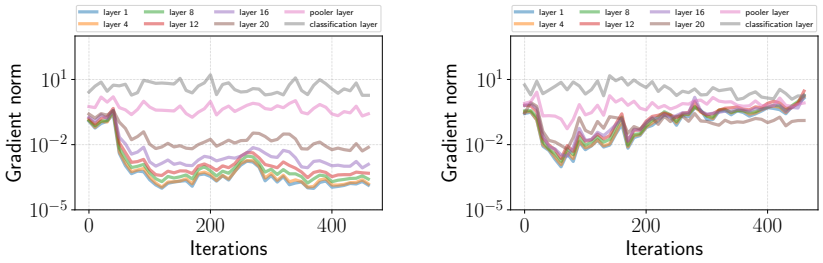
[1] BERT was trained on English Wikipedia, hence WikiText-2 can be seen as a subset of its training data.

[2] We remark that a similar experiment was done in Phang et al. (2018), but with a different goal of showing that their extended pre-training procedure is able to improve the fine-tuning stability.



(a) MRPC (b) CoLA

Figure 3: Results on down-sampled MRPC and CoLA using the default fine-tuning scheme of BERT (Devlin et al., 2019). The leftmost boxplot in each sub-figure shows the development accuracy when training on the full training set. Training on less samples hurts generalization but not stability.



(a) Failed run (b) Successful run

Figure 4: Gradient norms (plotted on a *logarithmic scale*) of different layers on RTE for a failed and sucessful run of BERT fine-tuning. We observe that the failed run is characterized by *vanishing gradients* in the bottom layers of the network. Additional plots for other weight matrices can be found in the appendix.

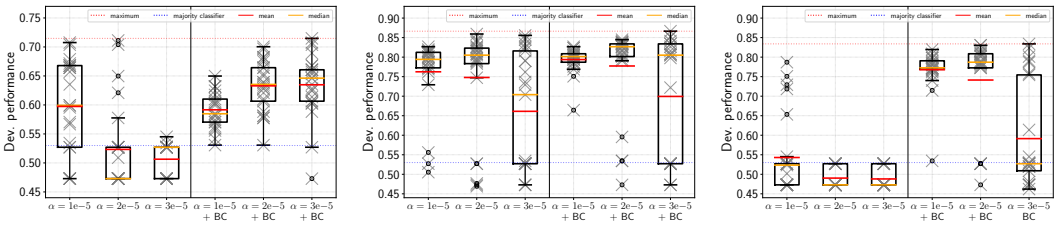
We conclude from this experiment, that the role of training dataset size per se is *orthogonal* to fine-tuning stability. What is crucial is rather for how many iterations we train. As our experiment shows, the observed increase in instability when training with smaller datasets can rather be attributed to the reduction of the number of iterations (changing the effective learning rate schedule) which, as we will show in the next section, has a crucial influence on the fine-tuning stability.

5 Disentangling optimization and generalization in fine-tuning instability

Our findings in Section 4 detail that while both catastrophic forgetting and small size of the datasets indeed *correlate* with fine-tuning instability, none of them are causing it. In this section, we argue that the fine-tuning instability is an optimization problem, and it admits a simple solution. Additionally, we show that even though a large fraction of the fine-tuning instability can be explained by optimization, the remaining instability can be attributed to generalization issues where fine-tuning runs with the same training loss exhibit noticeable differences in validation performance.

5.1 The role of optimization

Failed fine-tuning runs suffer from vanishing gradients. We observed in Fig. 2c that the failed runs have practically constant training loss throughout training. In order to better understand the nature of this phenomenon, in Fig. 4 we plot the ℓ_2 gradient norms of the loss function with respect to different layers of BERT, for one failed and successful fine-tuning run. For the failed run we see large gradients only for the top layers and *vanishing gradients* for the bottom layers. This is in large contrast to the successful run. While we also observe small gradients in the beginning of training (until iteration 70), gradients start to grow as training continues. Moreover, at the end of fine-tuning we observe the gradient norms nearly $2\times$ orders of magnitude larger than that of the



(a) BERT (b) RoBERTa (c) ALBERT

Figure 5: Box plots showing the fine-tuning performance of (a) BERT, (b) RoBERTa, (c) ALBERT for different learning rates α with and without bias correction (BC) on RTE. For BERT and ALBERT, having bias correction leads to more stable results and allows to train using larger learning rates. For RoBERTa the effect is less pronounced but still visible.

failed run. Similar visualizations for additional layers and weights can be found in the appendix. We observe the same behaviour also for RoBERTa and ALBERT models, and the corresponding figures can be found in Appendix (Fig. 12 and 13).

Importantly, we note that the vanishing gradients we observe during fine-tuning are harder to resolve than the standard *vanishing gradient problem* (Hochreiter, 1991; Bengio et al., 1994). In particular, common weight initialization schemes (Glorot and Bengio, 2010; He et al., 2015) ensure that the pre-activations of each layer of the network have zero mean and unit variance in expectation. However, we cannot simply modify the weights of a pre-trained model to ensure this property since this would conflict the idea of using the pre-trained weights.

Importance of bias correction in ADAM. Following Devlin et al. (2019), subsequent works on fine-tuning BERT-based models use the ADAM optimizer (Kingma and Ba, 2014). A subtle detail of the fine-tuning scheme of Devlin et al. (2019) is that it *does not* include the bias correction in ADAM.

Kingma and Ba (2014) already describe the effect of the bias correction as to reduce the learning rate at the beginning of training. By rewriting the update equations of ADAM as follows, we can clearly see this effect of bias correction:

$$\alpha_t \leftarrow \alpha \cdot \sqrt{1-\beta_2^t}/(1-\beta_1^t), \tag{1}$$

$$\theta_t \leftarrow \theta_{t-1} - \alpha_t \cdot m_t/(\sqrt{v_t}+\epsilon), \tag{2}$$

where m_t and v_t are biased first and second moment estimates respectively. Equation (1) shows that bias correction simply boils down to reducing the original step size α by a multiplicative factor $\sqrt{1-\beta_2^t}/(1-\beta_1^t)$ which is significantly below 1 for the first iterations of training and approaches 1 as the number of training iterations t increases (see Fig. 6). Along the same lines, You et al. (2020) explicitly remark that bias correction in ADAM has similar effect to warmup which is widely used in deep learning to prevent divergence early in training (He et al., 2016; Goyal et al., 2017; Devlin et al., 2019; Wong et al., 2020).

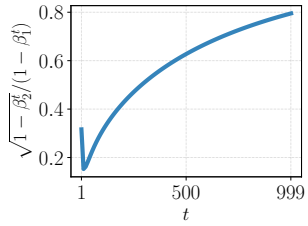
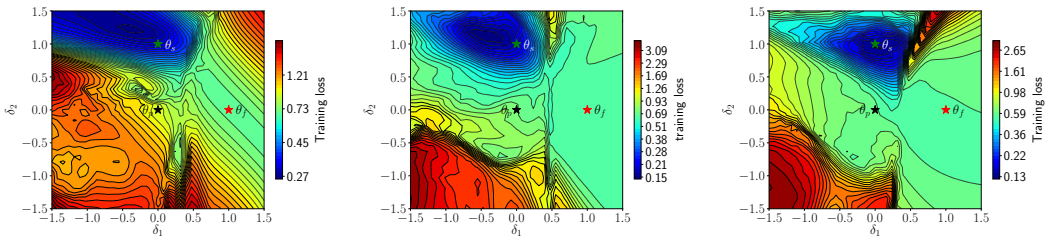


Figure 6: The bias correction term of ADAM ($\beta_1 = 0.9$ and $\beta_2 = 0.999$).

The implicit warmup of ADAM is likely to be an important factor that contributed to its success. We argue that fine-tuning BERT-based language models is not an exception. In Fig. 5 we show the results of fine-tuning on RTE with and without bias correction for BERT, RoBERTa, and ALBERT models. We observe that there is a significant benefit in combining warmup with bias correction, particularly for BERT and ALBERT. Even though for RoBERTa fine-tuning is already more stable even without bias correction, adding bias correction gives an additional improvement.[3]

Our results show that bias correction is useful if we want to get best performance within 3 epochs, the default recommendation by Devlin et al. (2019). An alternative solution is to simply train longer with a smaller learning rate, which also leads to much more stable fine-tuning. We provide a more detailed

[3]We note that bias correction is enabled by default in original fairseq (Ott et al., 2019) implementation of RoBERTa.



(a) RTE (b) MRPC (c) CoLA

Figure 7: 2D loss surfaces in the subspace spanned by $\delta_1 = \theta_f - \theta_p$ and $\delta_2 = \theta_s - \theta_p$ on RTE, MRPC and CoLA. $\theta_p, \theta_f, \theta_s$ denote the parameters of the pre-trained, failed, and successfully trained model, respectively.

ablation study in Appendix (Fig. 9) with analogous box plots for BERT using various learning rates, numbers of training epochs, with and without bias correction.

Loss surfaces. To get further intuition about the fine-tunung failure, we provide loss surface visualizations (Li et al., 2018; Hao et al., 2019) of failed and successful runs when fine-tuning BERT. Denote by $\theta_p, \theta_f, \theta_s$ the parameters of the pre-trained model, failed model, and successfully trained model, respectively. We plot a two-dimensional loss surface $f(\alpha, \beta) = \mathcal{L}(\theta_p + \alpha\delta_1 + \beta\delta_2)$ in the subspace spanned by $\delta_1 = \theta_f - \theta_p$ and $\delta_2 = \theta_s - \theta_p$ centered at the weights of the pre-trained model θ_p. Additional details are specified in the appendix.

Contour plots of the loss surfaces for RTE, MRPC, and CoLA are shown in Fig. 7. They provide further evidence for our findings on vanishing gradients: for failed fine-tuning runs gradient descent converges to a "bad" valley with a sub-optimal training loss. Moreover, this bad valley is separated from the local minimum (to which the successfully trained run converged) by a barrier. Interestingly, we observe a highly similar geometry for all three datasets providing further support for our interpretation of fine-tuning instability as a primarily optimization issue.

5.2 The role of generalization

Having established a better understanding of the fine-tuning optimization process and how it relates to the observed instability, we now turn to the generalizations aspects of fine-tuning instability. In order to show that the remaining fine-tuning variance can be attributed to generalization, we perform the following experiment. We fine-tune BERT on RTE following the default scheme of Devlin et al. (2019) but train for 10 epochs.

Fig. 8a shows the development set accuracy during fine-tuning for 10 successful runs. Fig. 8b shows the development set accuracy vs. training loss of *all* BERT models fine-tuned on RTE for this paper, in total 500 models (see also Fig. 9).

We find that despite achieving close to zero training loss overfitting is not an issue during fine-tuning. This is consistent with previous work by Hao et al. (2019), which arrived at a similar conclusion. Based on our results, we argue that it is even desirable to perform a large number of training iterations since the development accuracy varies considerably during fine-tuning and it does not degrade even when the training loss is as low as 10^{-5}.

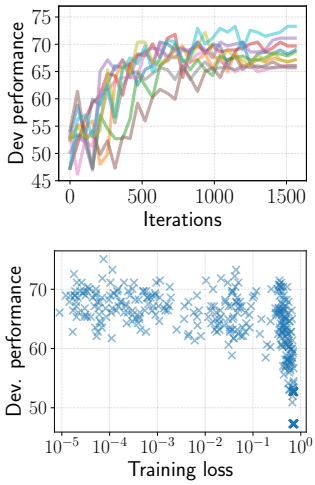


Figure 8: (a) Development accuracy on RTE during training. (b) Development accuracy vs. training loss at the end of training.

Combining these findings with results from the previous section we arrive at the conclusion that the observed fine-tuning instability can be decomposed into two aspects: optimization and generalization. In the next section we propose a simple solution addressing both issues.

Table 1: Standard deviation, mean, median, and maximum performance on the development set of RTE, MRPC and CoLA when fine-tuning BERT over 25 random seeds. Standard deviation: lower is better.

Approach	RTE				MRPC				CoLA			
	std	mean	median	max	std	mean	median	max	std	mean	median	max
Devlin et al. (2019)	7.2	52.3	47.3	71.1	5.9	79.0	74.8	89.4	25.6	45.6	59.1	64.6
Lee et al. (2020)	7.9	65.3	69.1	74.4	3.8	87.8	**89.5**	**91.8**	20.9	51.9	60.3	64.0
Ours	**2.4**	**70.1**	**70.0**	**75.5**	**0.8**	**89.6**	**89.5**	91.4	**1.0**	**64.6**	**64.7**	**66.8**

6 A simple but hard-to-beat baseline for fine-tuning BERT

As our findings in Section 5 show, the empirically observed instability of fine-tuning BERT can be attributed to vanishing gradients early in training as well as differences in generalization late in training. Given the new understanding of fine-tuning instability we propose the following guidelines for fine-tuning transformer-based masked language models:

- Use small learning rates combined with bias correction to avoid vanishing gradients early in training.
- Increase the number of iterations considerably and train to (almost) zero training loss while making use of early stopping.

Following our guidelines we propose a new baseline strategy for fine-tuning BERT: We use a learning rate of $2\mathrm{e}{-5}$ and train for 20 epochs. The learning rate is linearly increased for the first 10% of steps and linearly decayed to zero afterwards. We use early stopping based on validation accuracy. All other hyperparamters are kept unchanged. A full ablation study on all three dataset testing various combinations of the changed hyperparamters is presented in Section 7.3 in the appendix.

Results. Despite the simplicity of our proposed fine-tuning strategy we obtain strong empirical performance. Table 1 and Fig.1 show the results of fine-tuning BERT on RTE, MRPC, and CoLA. We compare to the default strategy by Devlin et al. (2019) and the recently proposed Mixout method by Lee et al. (2020). On RTE and CoLA, we do not only significantly improve fine-tuning stability ($3\times$ and $20\times$ smaller standard deviation, respectively) but also consistently improve overall performance (larger mean, median and maximum). On MRPC, we significantly improve fine-tuning stability ($4\times$ smaller standard deviation) while reaching overall similar performance in terms of mean, median and maximum score.

Finally, we note that the increased computational cost of our proposed fine-tuning scheme is not a major problem in practice. Comparing runtimes – on RTE, MRPC, and CoLA – to training for only 3 epochs, our fine-tuning scheme results in an average increase of training time by a factor of 6. Nevertheless we believe that overall our findings will lead to *more efficient* fine-tuning because of the significantly improved stability, effectively reducing the number of necessary fine-tuning runs.

7 Conclusions

In this work, we have resolved existing misconceptions about the reasons behind fine-tuning instability and proposed a new baseline strategy for fine-tuning that leads to significantly improved fine-tuning stability and overall improved results on three datasets from the GLUE benchmark.

By analyzing failed fine-tuning runs, we find that neither catastrophic forgetting nor small dataset sizes sufficiently explain fine-tuning instability. Rather, our analysis reveals that fine-tuning instability can be characterized by two distinct problems: (1) optimization difficulties early in training, characterized by vanishing gradients, and (2) differences in generalization, characterized by a large variation of development set accuracy for runs with almost equivalent training performance.

Based upon our analysis, we propose a simple but strong baseline strategy for fine-tuning BERT, even outperforming previous work in terms of fine-tuning stability and overall performance.

Broader Impact

Given the wide adoption and ubiquity of the pre-train then fine-tune scheme in today's NLP landscape, we believe that a better understanding of the fine-tuning process of pre-trained transformer-based language models can help to reduce the large computational cost of NLP models (Schwartz et al., 2019). Particularly, a better understanding of the fine-tuning instability combined with solutions for addressing it helps to avoid the need to repeat fine-tuning experiments a large number of times in order to obtain the best score on a leaderboard.

Acknowledgments

We thank Anna Khokhlova for her help with the language modeling experiments, Cheolhyoung Lee and Jesse Dodge for providing us with details of their works, and Badr Abdullah and Aditya Mogadala for their helpful comments on a draft of this paper.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project-id 232722074 – SFB 1102.

References

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, and Danilo Giampiccolo. 2006. The second pascal recognising textual entailment challenge. *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *In Proceedings of the Second Text Analysis Conference (TAC 2009)*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, page 177–190, Berlin, Heidelberg. Springer-Verlag.

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.

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:2002.06305*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

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*, RTE '07, page 1–9, USA. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of BERT. 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 4143–4152, Hong Kong, China. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1).

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

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

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6389–6399. Curran Associates, Inc.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*.

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*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R Bowman. 2020. Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work? *arXiv preprint arXiv:2005.00628*.

R Schwartz, J Dodge, NA Smith, et al. 2019. Green ai (2019). *arXiv preprint arXiv:1907.10597*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3266–3280. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Eric Wong, Leslie Rice, and J. Zico Kolter. 2020. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. *arXiv preprint arXiv:2002.04745*.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*.

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

Appendix

7.1 Task statistics

Statistics for each of the datasets studied in this paper. All datasets are publicly available and can be downloaded here: `https://github.com/nyu-mll/jiant`.

Table 2: Dataset statistics and majority baselines.

	RTE	**MRPC**	**CoLA**
Training	2491	3669	8551
Development	278	409	1043
Majority baseline	0.53	0.75	0.0
Metric	Acc.	$\frac{F_1+Acc.}{2}$	MCC

7.2 Hyperparamters

Hyperparameters for BERT, RoBERTa, and ALBERT used for all our experiments.

Table 3: Hyperparameters used for fine-tuning.

Hyperparam	**BERT**	**RoBERTa**	**ALBERT**
Epochs	$3, 10, 20$	3	3
Learning rate	$1e{-}5 - 5e{-}5$	$1e{-}5 - 3e{-}5$	$1e{-}5 - 3e{-}5$
Learning rate schedule	warmup-linear	warmup-linear	warmup-linear
Warmup ratio	0.1	0.1	0.1
Batch size	16	16	16
Adam ϵ	$1e{-}6$	$1e{-}6$	$1e{-}6$
Adam β_1	0.9	0.9	0.9
Adam β_2	0.999	0.98	0.999
Adam bias correction	{True, False}	{True, False}	{True, False}
Dropout	0.1	0.1	–
Weight decay	0.01	0.1	–
Clipping gradient norm	1.0	–	1.0
Number of random seeds	25	25	25

7.3 Ablation studies

Figures 9, 10, and 11 show the results of fine-tuning on RTE, MRPC, and CoLA with different combinations of learning rate, number of training epochs, and bias correction. We make the following observations:

- When training for only 3 epochs, disabling bias correction clearly hurts performance.
- With bias correction, training with larger learning rates is possible.
- Combining the usage of bias correction with training for more epochs leads to the best performance.

7.4 Additional gradient norm visualizations

We provide additional visualizations for the vanishing gradients observed when fine-tuning RoBERTa, ALBERT, and BERT in Figures 12, 13, 14. Note that for ALBERT besides the pooler and classification layers, we plot only the gradient norms of a single hidden layer (referred to as `layer0`) because of weight sharing.

Gradient norms and MLM perplexity. We can see from the gradient norm visualizations for BERT in Figures 4 and 14 that the gradient norm of the pooler and classification layer remains large. Hence,

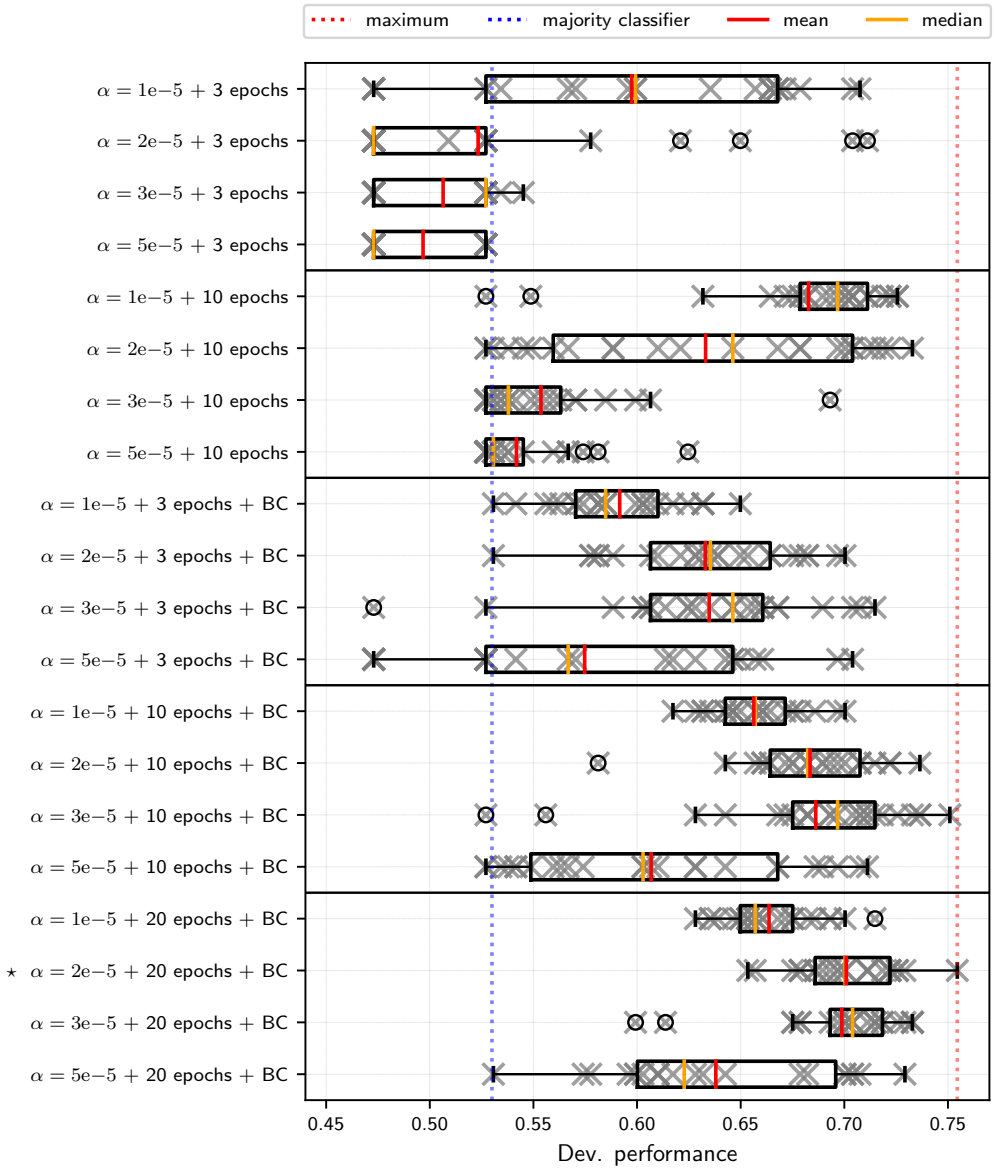


Figure 9: Full ablation of fine-tuning BERT on **RTE**. For each setting we vary only the number of training steps, learning rate, and usage of bias correction (BC). All other hyperparameters are unchanged. We fine-tune 25 models for each setting. ⋆ shows the setting which we recommend as a new baseline fine-tuning strategy.

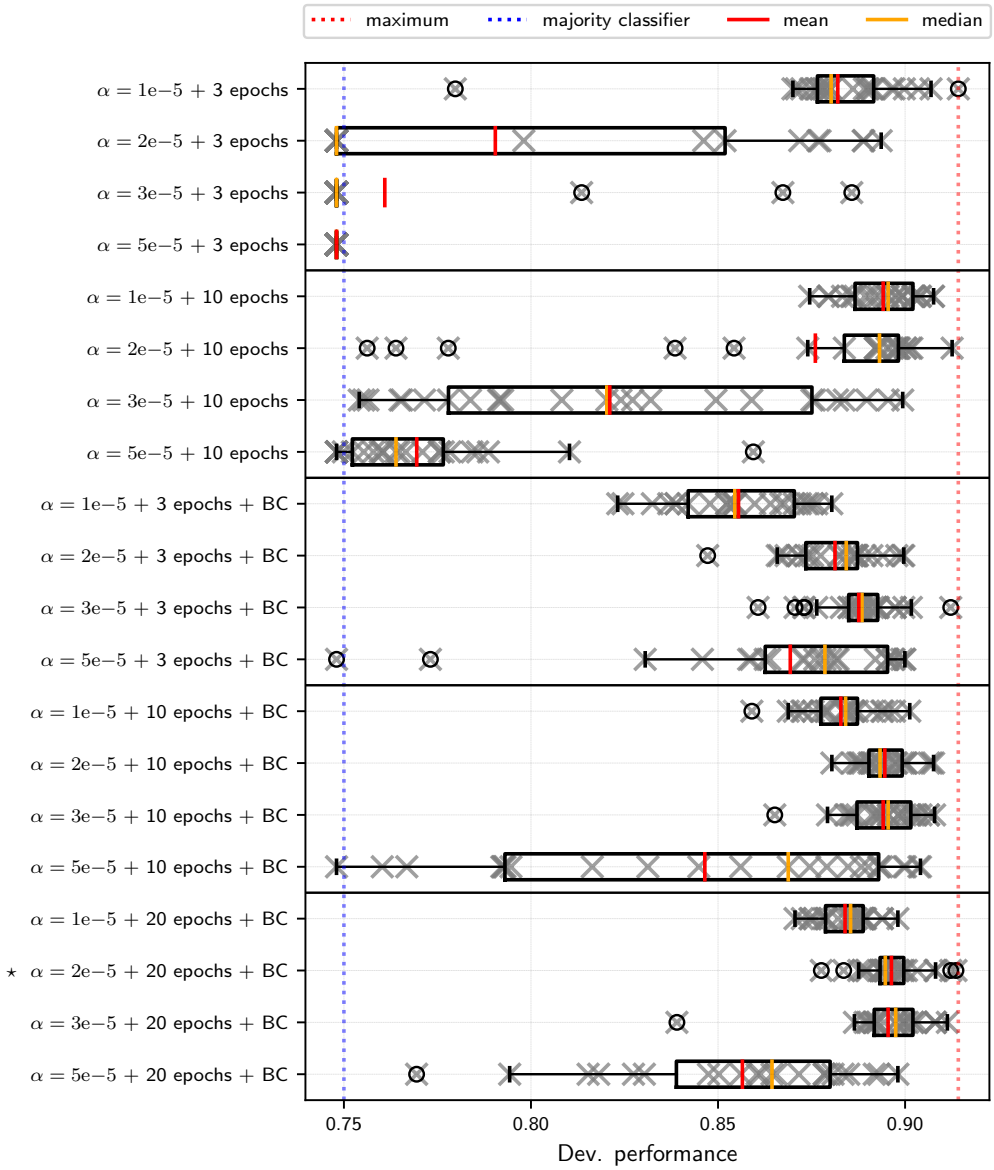


Figure 10: Full ablation of fine-tuning BERT on **MRPC**. For each setting we vary only the number of training steps, learning rate, and usage of bias correction (BC). All other hyperparameters are unchanged. We fine-tune 25 models for each setting. ⋆ shows the setting which we recommend as a new baseline fine-tuning strategy.

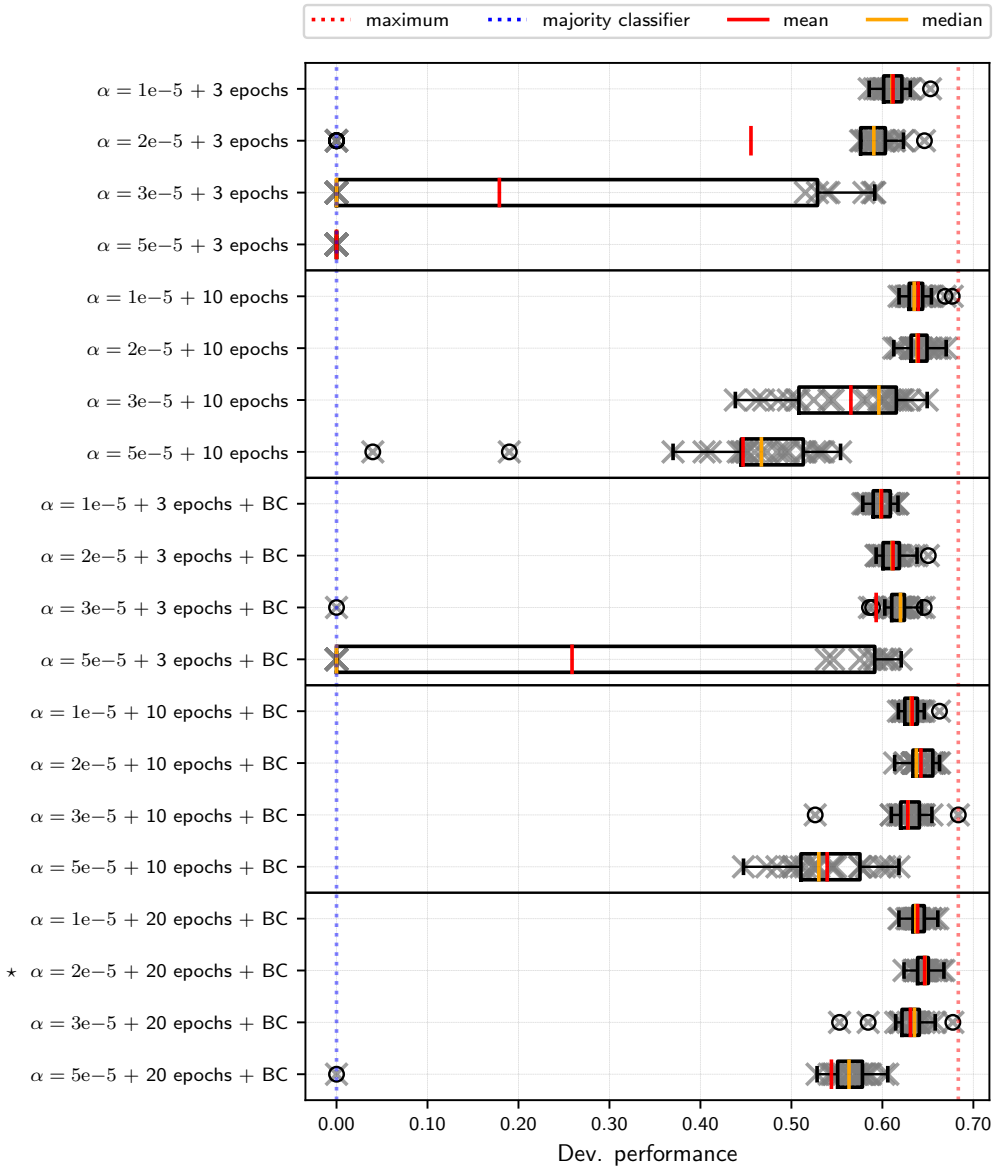


Figure 11: Full ablation of fine-tuning BERT on **CoLA**. For each setting we vary only the number of training steps, learning rate, and usage of bias correction (BC). All other hyperparameters are unchanged. We fine-tune 25 models for each setting. ⋆ shows the setting which we recommend as a new baseline fine-tuning strategy.

even though the gradients on most layers of the model vanish, we still update the weights on the top layers. In fact, this explains the large increase in MLM perplexity for the failed models which is shown in Fig. 2a. While most of the layers do not change as we continue training, the top layers of network change dramatically.

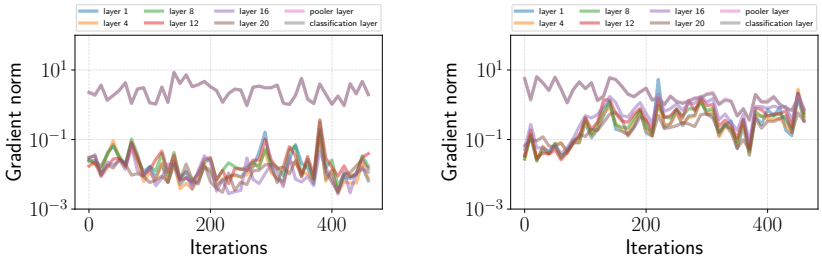
7.5 Loss surfaces

For Fig. 7, we define the range for both α and β as $[-1.5, 1.5]$ and sample 40 points for each axis. We evaluate the loss on 128 samples from the training dataset of each task using *all* model parameters, including the classification layer. We disabled dropout for generating the surface plots.

Fig. 15 shows contour plots of the total gradient norm. We can again see that the point to which the failed model converges to (θ_f) is separated from the point the successful model converges to (θ_s) by a barrier. Moreover, on all the three datasets we can clearly see the valley around θ_f with a small gradient norm.

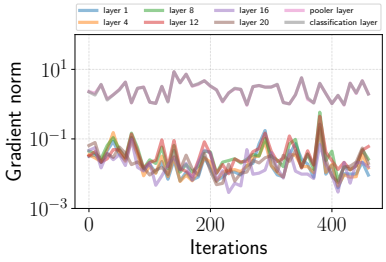
7.6 Generalization

We provide additional plots of training loss versus validation performance for MRPC and CoLA in Fig. 16 (we also show the plot for RTE again for a side-by-side comparison). We use the 500 models from the ablation study (see Fig. 9, 10, 11). We observe a significant variance in validation performance even though the training loss is the same. This occurs not only on RTE, but on MRPC and CoLA as well which have larger validation sets (see Table 2). Moreover, we can observe that there is no overfitting when training to close to zero training loss. This justifies our fine-tuning scheme which involves training for a larger number of epochs.

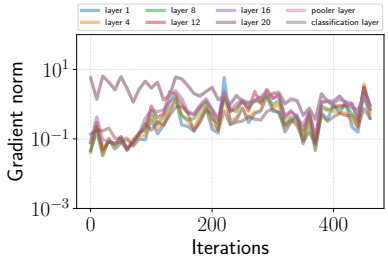


(a) Failed run: attention.key

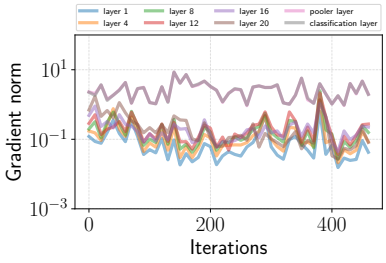
(b) Successful run: attention.key



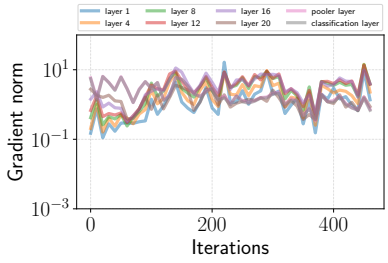
(c) Failed run: attention.query



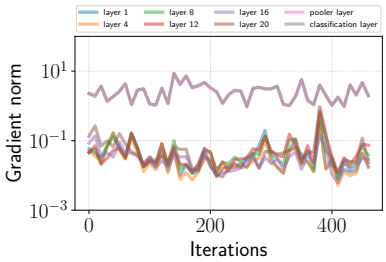
(d) Successful run: attention.query



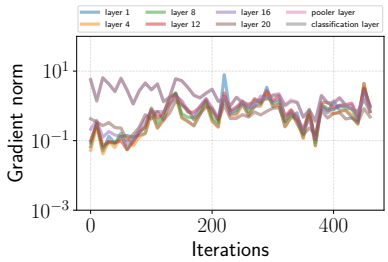
(e) Failed run: attention.value



(f) Successful run: attention.value

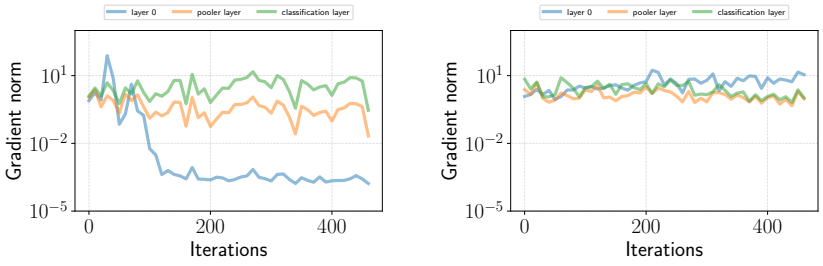


(g) Failed run: attention.output.dense



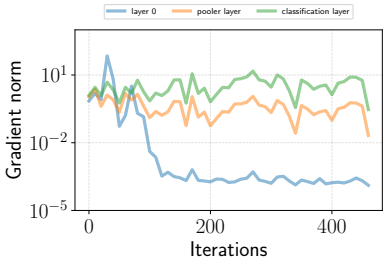
(h) Successful run: attention.output.dense

Figure 12: Gradient norms (plotted on a *logarithmic scale*) of additional weight matrices of **RoBERTa** fine-tuned on RTE. Corresponding layer names are in the captions. We show gradient norms corresponding to a single failed and single successful, respectively.

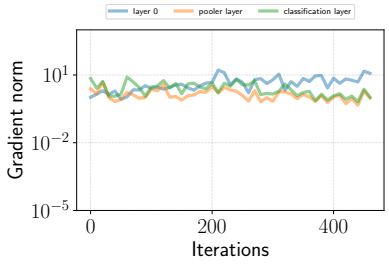


(a) Failed run: attention.key

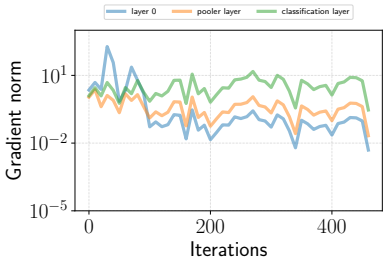
(b) Successful run: attention.key



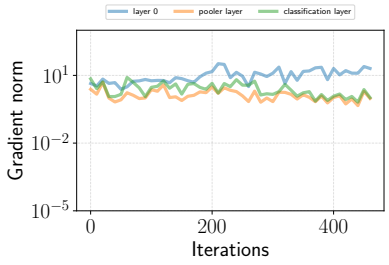
(c) Failed run: attention.query



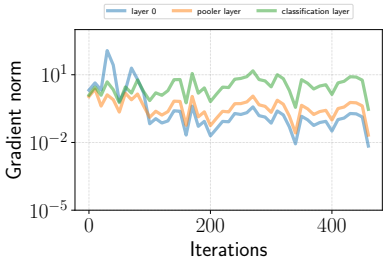
(d) Successful run: attention.query



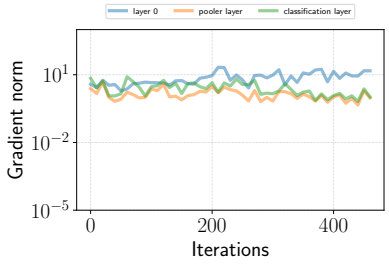
(e) Failed run: attention.value



(f) Successful run: attention.value

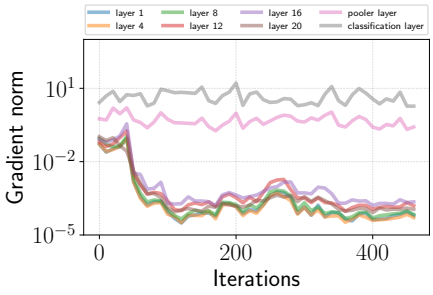


(g) Failed run: attention.dense

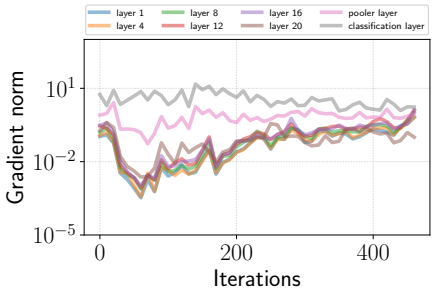


(h) Successful run: attention.dense

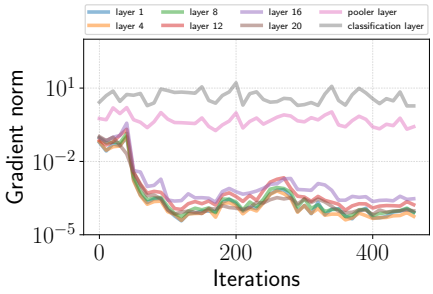
Figure 13: Gradient norms (plotted on a *logarithmic scale*) of additional weight matrices of **ALBERT** fine-tuned on RTE. Corresponding layer names are in the captions. We show gradient norms corresponding to a single failed and single successful, respectively.



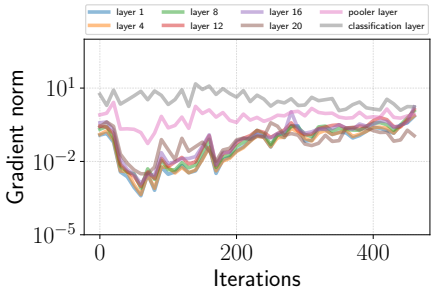
(a) Failed run: attention.key



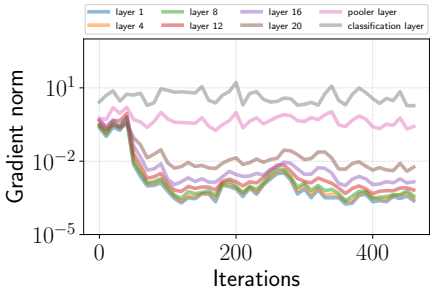
(b) Successful run: attention.key



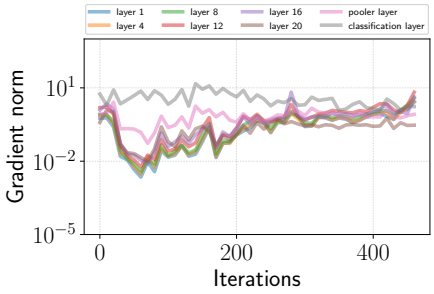
(c) Failed run: attention.query



(d) Successful run: attention.query

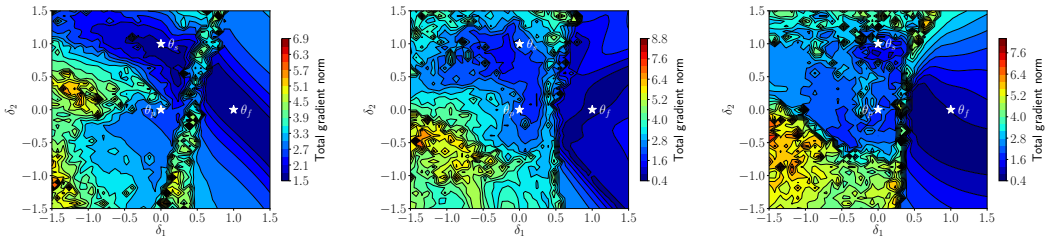


(e) Failed run: attention.value



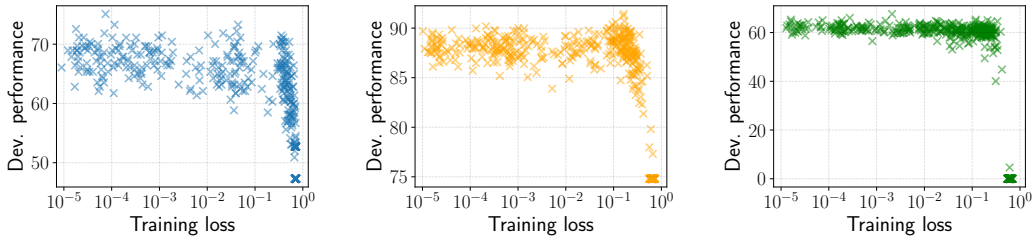
(f) Successful run: attention.value

Figure 14: Gradient norms (plotted on a *logarithmic scale*) of additional weight matrices of **BERT** fine-tuned on RTE. Corresponding layer names are in the captions. We show gradient norms corresponding to a single failed and single successful, respectively.



(a) RTE (b) MRPC (c) CoLA

Figure 15: 2D gradient norm surfaces in the subspace spanned by $\delta_1 = \theta_f - \theta_p$ and $\delta_2 = \theta_s - \theta_p$ for BERT fine-tuned on RTE, MRPC and CoLA. $\theta_p, \theta_f, \theta_s$ denote the parameters of the pre-trained, failed, and successfully trained model, respectively.



(a) RTE (b) MRPC (c) CoLA

Figure 16: Development performance versus training loss at the end of training for 500 models fine-tuned on RTE, MRPC, and CoLA, respectively.