Visualizing and Understanding the Effectiveness of BERT

Yaru Hao[†][*], **Li Dong**[‡], **Furu Wei**[‡], **Ke Xu**[†]
[†]Beihang University
[‡]Microsoft Research
{haoyaru@,kexu@nlsde.}buaa.edu.cn
{lidong1,fuwei}@microsoft.com

Abstract

Language model pre-training, such as BERT, has achieved remarkable results in many NLP tasks. However, it is unclear why the pre-training-then-fine-tuning paradigm can improve performance and generalization capability across different tasks. In this paper, we propose to visualize loss landscapes and optimization trajectories of fine-tuning BERT on specific datasets. First, we find that pre-training reaches a good initial point across downstream tasks, which leads to wider optima and easier optimization compared with training from scratch. We also demonstrate that the fine-tuning procedure is robust to overfitting, even though BERT is highly over-parameterized for downstream tasks. Second, the visualization results indicate that fine-tuning BERT tends to generalize better because of the flat and wide optima, and the consistency between the training loss surface and the generalization error surface. Third, the lower layers of BERT are more invariant during fine-tuning, which suggests that the layers that are close to input learn more transferable representations of language.

1 Introduction

Language model pre-training has achieved strong performance in many NLP tasks (Peters et al., 2018; Howard and Ruder, 2018a; Radford et al., 2018; Devlin et al., 2018; Baevski et al., 2019; Dong et al., 2019). A neural encoder is trained on a large text corpus by using language modeling objectives. Then the pre-trained model either is used to extract vector representations for input, or is fine-tuned on the specific datasets.

Recent work (Tenney et al., 2019b; Liu et al., 2019a; Goldberg, 2019; Tenney et al., 2019a) has shown that the pre-trained models can encode syntactic and semantic information of language. However, it is unclear why pre-training is effective on downstream tasks in terms of both trainability and generalization capability. In this work, we take BERT (Devlin et al., 2018) as an example to understand the effectiveness of pre-training. We visualize the loss landscapes and the optimization procedure of fine-tuning on specific datasets in three ways. First, we compute the one-dimensional (1D) loss curve, so that we can inspect the difference between fine-tuning BERT and training from scratch. Second, we visualize the two-dimensional (2D) loss surface, which provides more information about loss landscapes than 1D curves. Third, we project the high-dimensional optimization trajectory of fine-tuning to the obtained 2D loss surface, which demonstrate the learning properties in an intuitive way.

The main findings are summarized as follows. First, visualization results indicate that BERT pre-training reaches a good initial point across downstream tasks, which leads to wider optima on the 2D loss landscape compared with random initialization. Moreover, the visualization of optimization trajectories shows that pre-training results in easier optimization and faster convergence. We also demonstrate that the fine-tuning procedure is robust to overfitting. Second, loss landscapes of fine-tuning partially explain the good generalization capability of BERT. Specifically, pre-training obtains more flat and wider optima, which indicates the pre-trained model tends to generalize better on unseen data (Chaudhari et al., 2017; Li et al., 2018; Izmailov et al., 2018). Additionally, we find that the training loss surface correlates well with the generalization error. Third, we demonstrate that the lower (i.e., close to input) layers of BERT are more invariant across tasks than the higher layers, which suggests that the lower layers learn transferable representations of language. We verify the point by visualizing the loss landscape with respect to different groups of layers.

[*]Contribution during internship at Microsoft Research.

2 Background: BERT

We use BERT (Bidirectional Encoder Representations from Transformers; Devlin et al. 2018) as an example of pre-trained language models in our experiments. BERT is pre-trained on a large corpus by using the masked language modeling and next-sentence prediction objectives. Then we can add task-specific layers to the BERT model, and fine-tune all the parameters according to the downstream tasks.

BERT employs a Transformer (Vaswani et al., 2017) network to encode contextual information, which contains multi-layer self-attention blocks. Given the embeddings $\{\mathbf{x}_i\}_{i=1}^{|x|}$ of input text, we concatenate them into $\mathbf{H}^0 = [\mathbf{x}_1, \cdots, \mathbf{x}_{|x|}]$. Then, an L-layer Transformer encodes the input: $\mathbf{H}^l = \text{Transformer\_block}_l(\mathbf{H}^{l-1})$, where $l = 1, \cdots, L$, and $\mathbf{H}^L = [\mathbf{h}_1^L, \cdots, \mathbf{h}_{|x|}^L]$. We use the hidden vector $\mathbf{h}_i^L$ as the contextualized representation of the input token x_i. For more implementation details, we refer readers to Vaswani et al. (2017).

3 Methodology

We employ three visualization methods to understand why fine-tuning the pre-trained BERT model can achieve better performance on downstream tasks compared with training from scratch. We plot both one-dimensional and two-dimensional loss landscapes of BERT on the specific datasets. Besides, we project the optimization trajectories of the fine-tuning procedure to the loss surface. The visualization algorithms can also be used for the models that are trained from random initialization, so that we can compare the difference between two learning paradigm.

3.1 One-dimensional Loss Curve

Let $\boldsymbol{\theta}_0$ denote the initialized parameters. For fine-tuning BERT, $\boldsymbol{\theta}_0$ represents the the pre-trained parameters. For training from scratch, $\boldsymbol{\theta}_0$ represents the randomly initialized parameters. After fine-tuning, the model parameters are updated to $\boldsymbol{\theta}_1$. The one-dimensional (1D) loss curve aims to quantify the loss values along the optimization direction (i.e., from $\boldsymbol{\theta}_0$ to $\boldsymbol{\theta}_1$).

The loss curve is plotted by linear interpolation between $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ (Goodfellow and Vinyals, 2015). The curve function $f(\alpha)$ is defined as:

$$f(\alpha) = \mathcal{J}(\boldsymbol{\theta}_0 + \alpha\boldsymbol{\delta}_1) \tag{1}$$

where α is a scalar parameter, $\boldsymbol{\delta}_1 = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_0$ is the optimization direction, and $\mathcal{J}(\boldsymbol{\theta})$ is the loss function under the model parameters $\boldsymbol{\theta}$. In our experiments, we set the range of α to $[-4, 4]$ and sample 40 points for each axis. Note that we only consider the parameters of BERT in $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$, so $\boldsymbol{\delta}_1$ only indicates the updates of the original BERT parameters. The effect of the added task-specific layers is eliminated by keeping them fixed to the learned values.

3.2 Two-dimensional Loss Surface

The one-dimensional loss curve can be extended to the two-dimensional (2D) loss surface (Li et al., 2018). Similar as in Equation (1), we need to define two directions ($\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$) as axes to plot the loss surface:

$$f(\alpha, \beta) = \mathcal{J}(\boldsymbol{\theta}_0 + \alpha\boldsymbol{\delta}_1 + \beta\boldsymbol{\delta}_2) \tag{2}$$

where α, β are scalar values, $\mathcal{J}(\cdot)$ is the loss function, and $\boldsymbol{\theta}_0$ represents the initialized parameters. Similar to Section 3.1, we are only interested in the parameter space of the BERT encoder, without taking into consideration task-specific layers. One of the axes is the optimization direction $\boldsymbol{\delta}_1 = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_0$ on the target dataset, which is defined in the same way as in Equation (1). We compute the other axis direction via $\boldsymbol{\delta}_2 = \boldsymbol{\theta}_2 - \boldsymbol{\theta}_0$, where $\boldsymbol{\theta}_2$ represents the fine-tuned parameters on another dataset. So the other axis is the optimization direction of fine-tuning on another dataset. Even though the other dataset is randomly chosen, experimental results confirm that the optimization directions $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2$ are divergent and orthogonal to each other because of the high-dimensional parameter space.

The direction vectors $\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$ are projected onto a two-dimensional plane. It is beneficial to ensure the scale equivalence of two axes for visualization purposes. Similar to the filter normalization approach introduced in (Li et al., 2018), we address this issue by normalizing two direction vectors to the same norm. We re-scale $\boldsymbol{\delta}_2$ to $\frac{\|\boldsymbol{\delta}_1\|}{\|\boldsymbol{\delta}_2\|}\boldsymbol{\delta}_2$, where $\|\cdot\|$ denotes the Euclidean norm. We set the range of both α and β to $[-4, 4]$ and sample 40 points for each axis.

3.3 Optimization Trajectory

Our goal is to project the optimization trajectory of the fine-tuning procedure onto the 2D loss surface obtained in Section 3.2. Let $\{(d_i^\alpha, d_i^\beta)\}_{i=1}^T$ denote the projected optimization trajectory, where

(d_i^α, d_i^β) is a projected point in the loss surface, and $i = 1, \cdots, T$ represents the i-th epoch of fine-tuning.

As shown in Equation (2), we have known the optimization direction $\boldsymbol{\delta}_1 = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_0$ on the target dataset. We can compute the deviation degrees between the optimization direction and the trajectory to visualize the projection results. Let $\boldsymbol{\theta}^i$ denote the BERT parameters at the i-th epoch, and $\boldsymbol{\delta}^i = \boldsymbol{\theta}^i - \boldsymbol{\theta}_0$ denote the optimization direction at the i-th epoch. The point (d_i^α, d_i^β) of the trajectory is computed via:

$$v_{cos} = \frac{\boldsymbol{\delta}^i \times \boldsymbol{\delta}_1}{\|\boldsymbol{\delta}^i\| \cdot \|\boldsymbol{\delta}_1\|} \tag{3}$$

$$d_i^\alpha = v_{cos} \frac{\|\boldsymbol{\delta}^i\|}{\|\boldsymbol{\delta}_1\|} = \frac{\boldsymbol{\delta}^i \times \boldsymbol{\delta}_1}{\|\boldsymbol{\delta}_1\|^2} \tag{4}$$

$$d_i^\beta = \sqrt{(\frac{\|\boldsymbol{\delta}^i\|}{\|\boldsymbol{\delta}_1\|})^2 - (d_i^\alpha)^2} \tag{5}$$

where $\times$ denotes the cross product of two vectors, and $\|\cdot\|$ denotes the Euclidean norm. To be specific, we first compute cosine similarity between $\boldsymbol{\delta}^i$ and $\boldsymbol{\delta}_1$, which indicates the angle between the current optimization direction and the final optimization direction. Then we get the projection values d_i^α and d_i^β by computing the deviation degrees between the optimization direction $\boldsymbol{\delta}^i$ and the axes.

4 Experimental Setup

We conduct experiments on four datasets: Multi-genre Natural Language Inference Corpus (MNLI; Williams et al. 2018), Recognizing Textual Entailment (RTE; Dagan et al. 2006; Bar-Haim et al. 2006; Giampiccolo et al. 2007; Bentivogli et al. 2009), Stanford Sentiment Treebank (SST-2; Socher et al. 2013), and Microsoft Research Paraphrase Corpus (MRPC; Dolan and Brockett 2005). We use the same data split as in (Wang et al., 2019). The accuracy metric is used for evaluation.

We employ the pre-trained BERT-large model in our experiments. The cased version of tokenizer is used. We follow the settings and the hyper-parameters suggested in (Devlin et al., 2018). The Adam (Kingma and Ba, 2015) optimizer is used for fine-tuning. The number of fine-tuning epochs is selected from $\{3, 4, 5\}$. For RTE and MRPC, we set the batch size to 32, and the learning rate to 1e-5. For MNLI and SST-2, the batch size is 64, and the learning rate is 3e-5.

For the setting of training from scratch, we use the same network architecture as BERT, and randomly initialize the model parameters. Most hyper-parameters are kept the same. The number of training epochs is larger than fine-tuning BERT, because training from scratch requires more epochs to converge. The number of epochs is set to 8 for SST-2, and 16 for the other datasets, which is validated on the development set.

5 Pre-training Gets a Good Initial Point Across Downstream Tasks

Fine-tuning BERT on the usually performs significantly better than training the same network with random initialization, especially when the data size is small. Results indicate that language model pre-training objectives learn good initialization for downstream tasks. In this section, we inspect the benefits of using BERT as the initial point from three aspects.

5.1 Pre-training Leads to Wider Optima

As described in Section 3.2, we plot 2D training loss surfaces on four datasets in Figure 1. The figures in the top row are the models trained from scratch, while the others in bottom are based on fine-tuning BERT. The start point indicates the training loss of the initialized model, while the end point indicates the final training loss. We observe that the optima obtained by fine-tuning BERT are much wider than training from scratch. A wide optimum of fine-tuning BERT implicates that the small perturbations of the model parameters cannot hurt the final performance seriously, while a thin optimum is more sensitive to these subtle changes. Moreover, in Section 6.1, we further discuss about the width of the optima can contribute to the generalization capability.

As shown in Figure 1, the fine-tuning path from the start point to the end point on the loss landscape is more smooth than training from scratch. In other words, the training loss of fine-tuning BERT tends to monotonously decrease along the optimization direction, which eases optimization and accelerates training convergence. In contrast, the path from random initial point to the end point is more rough, which requires a more carefully tweaked optimizer to obtain reasonable performance.

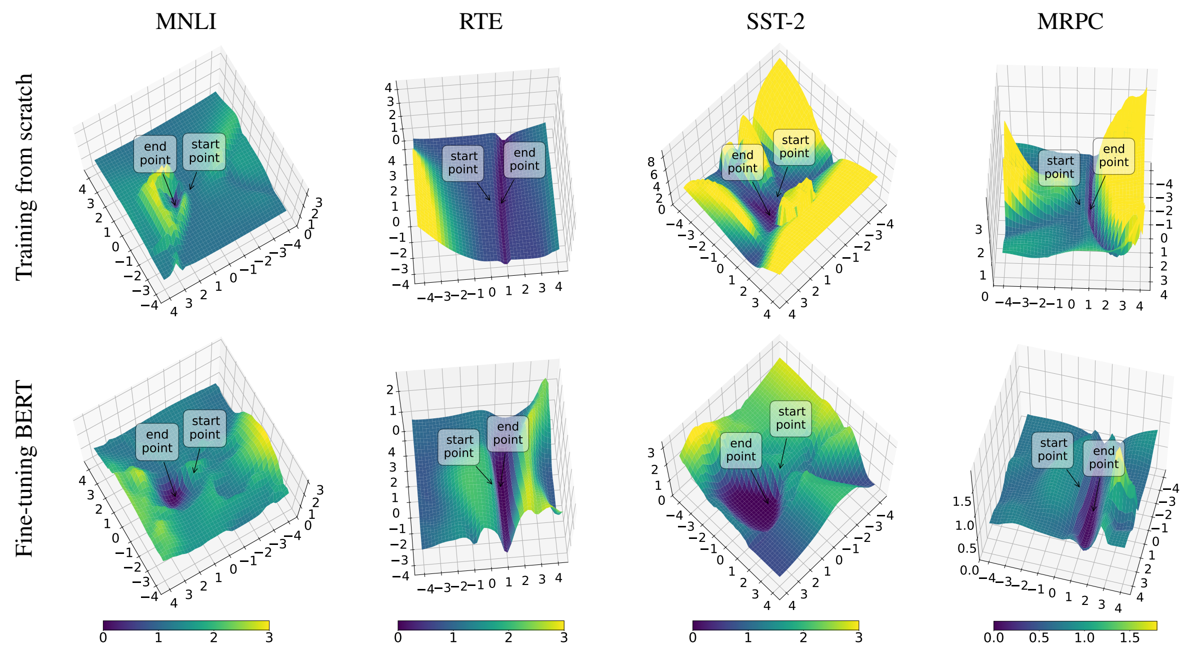

Figure 1: Training loss surfaces of training from scratch (top) and fine-tuning BERT (bottom) on four datasets. We annotate the start point (i.e., initialized model) and the end point (i.e., estimated model) in the loss surfaces. Pre-training leads to wider optima, and eases optimization compared with random initialization.

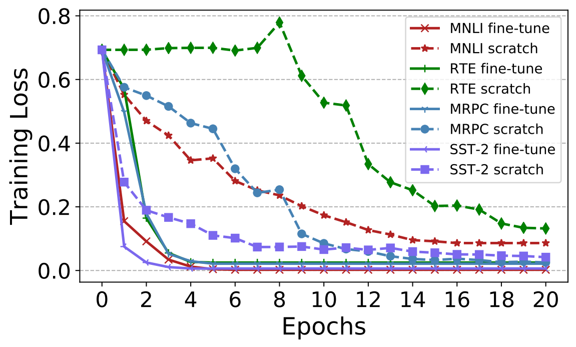

Figure 2: Training loss of fine-tuning BERT and training from scratch on four datasets.

5.2 Pre-training Eases Optimization on Downstream Tasks

We fine-tune BERT and train the same network from scratch on four datasets. The learning curves are shown in Figure 2. We find that training from scratch requires more iterations to converge on the datasets, while pre-training-then-fine-tuning converges faster in terms of training loss. We also notice that the final loss of training from scratch tends to be higher than fine-tuning BERT, even if it undergoes more epochs. On the RTE dataset, training the model from scratch has a hard time decreasing the loss in the first few epochs.

In order to visualize the dynamic convergence process, we plot the optimization trajectories using the method described in Section 3.3. As shown in Figure 3, for training from scratch, the optimization directions of the first few epochs are divergent from the final optimization direction. Moreover, the loss landscape from the initial point to the end point is more rough than fine-tuning BERT, we can see that the trajectory of training from scratch on the MRPC dataset crosses an obstacle to reach the end point.

Compared with training from scratch, fine-tuning BERT finds the optimization direction in a more straightforward way. The optimization process also converges faster. Besides, the fine-tuning path is unimpeded along the optimization direction. In addition, because of the wider optima near the initial point, fine-tuning BERT tends to reach the expected optimal region even if it optimizes along the direction of the first epoch.

5.3 Pre-training-then-fine-tuning is Robust to Overfitting

The BERT-large model has 345M parameters, which is over-parameterized for the target datasets. However, experimental results show fine-tuning BERT is robust to over-fitting, i.e., the generalization error (namely, the classification error rate on the development set) does not dramatically increase for more training epochs, despite the huge number of model parameters.

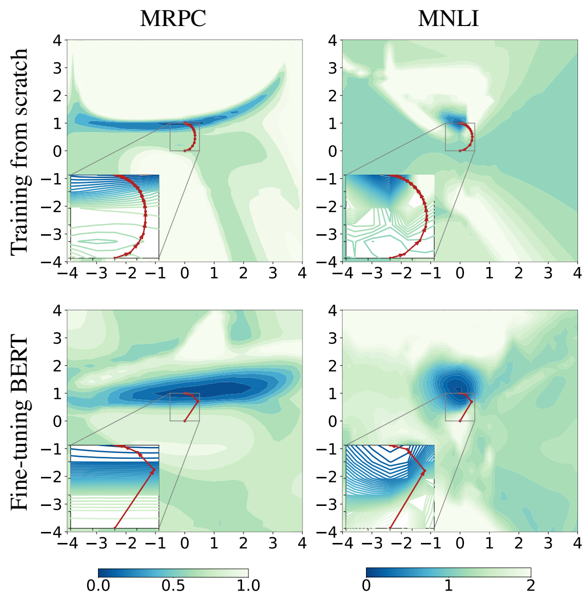

Figure 3: The optimization trajectories on the training loss surfaces of training from scratch (top) and fine-tuning BERT (bottom) on the MRPC and MNLI datasets. Darker color represents smaller training loss.

We use the MRPC dataset as a case study, because its data size is relatively small, which is prone to overfitting if we train the model from scratch. As shown in Figure 4, we plot the optimization trajectory of fine-tuning on the generalization error surface. We first fine-tune the BERT model for five epochs as suggested in (Devlin et al., 2018). Then we continue fine-tuning for another twenty epochs, which still obtains comparable performance with the first five epochs. Figure 4 shows that even though we fine-tune the BERT model for twenty more epochs, the final estimation is not far away from its original optimum. Moreover, the optimum area is wide enough to avoid the model from jumping out the region with good generalization capability, which explains why the pre-training-then-fine-tuning paradigm is robust to overfitting.

6 Pre-training Helps to Generalize Better

Although training from scratch can achieve comparable training losses as fine-tuning BERT, the model with random initialization usually has poor performance on the unseen data. In this section, we use visualization techniques to understand why the model obtained by pre-training-then-fine-tuning tends to have better generalization capability.

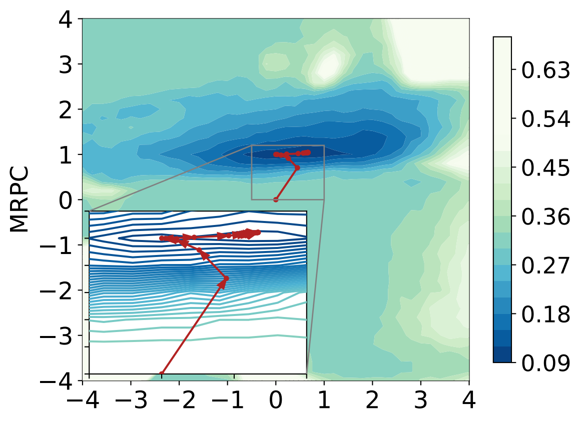

Figure 4: The optimization trajectory of fine-tuning $5 + 20$ epochs on MRPC. The two-dimensional generalization error surface is presented. We find that pre-training-then-fine-tuning is robust to overfitting.

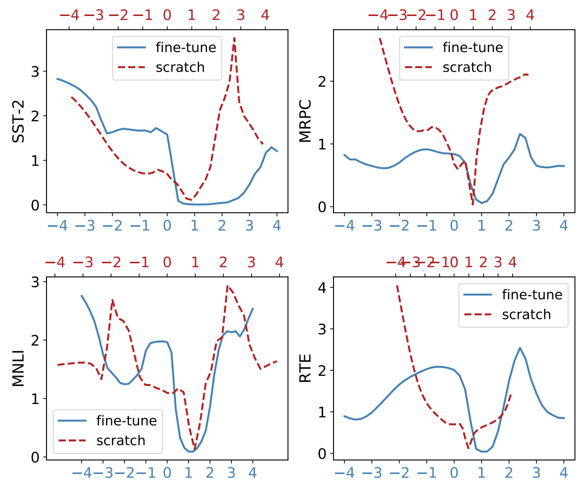

Figure 5: One-dimensional training loss curves. Dash lines represent training from scratch, and solid lines represent fine-tuning BERT. The scales of axes are normalized for comparison. The optima of training from scratch are sharper than fine-tuning BERT.

6.1 Wide and Flat Optima Lead to Better Generalization

Previous work (Hochreiter and Schmidhuber, 1997; Keskar et al., 2016; Li et al., 2018) shows that the flatness of a local optimum correlates with the generalization capability, i.e., more flat optima lead to better generalization. The finding inspires us to inspect the loss landscapes of BERT fine-tuning, so that we can understand the generalization capability from the perspective of the flatness of optima.

Section 5.1 presents that the optima obtained by fine-tuning BERT are wider than training from scratch. As shown in Figure 5, we further plot one-dimensional training loss curves of both fine-tuning BERT and training from scratch, which represents the transverse section of two-dimensional loss surface along the optimization direction. We normalize the scale of axes for flatness comparison as suggested in (Li et al., 2018). Figure 5 shows that the optima of fine-tuning BERT are more flat, while training from scratch obtains more sharp optima. The results indicate that pre-training-then-fine-tuning tends to generalize better on unseen data.

6.2 Consistency Between Training Loss Surface and Generalization Error Surface

To further understand the effects of geometry of loss functions to the generalization capability, we make comparisons between the training loss surfaces and the generalization error surfaces on different datasets. The classification error rate on the development set is used as an indicator of the generalization capability.

As shown in Figure 6, we find the end points of fine-tuning BERT fall into the wide areas with smaller generalization error. The results show that the generalization error surfaces are consistent with the corresponding training loss surfaces on the datasets, i.e., smaller training loss tends to decrease the error on the development set. Moreover, the fine-tuned BERT models tend to stay approximately optimal under subtle perturbations. The visualization results also indicate that it is preferred to converge to wider and more flat local optima, as the training loss surface and the generalization error surface are shifted with respect to each other (Izmailov et al., 2018). In contrast, training from scratch obtains thinner optimum areas and poorer generalization than fine-tuning BERT, especially on the datasets with relatively small data size (such as MRPC, and RTE). Intuitively, the thin and sharp optima on the training loss surfaces are hard to be migrated to the generalization surfaces.

For training from scratch, it is not surprising that on larger datasets (such as MNLI, and SST-2) the generalization error surfaces are more consistent with the training loss surfaces. The results suggest that training the model from scratch usually requires more training examples to generalize better compared with fine-tuning BERT.

7 Lower Layers of BERT are More Invariant and Transferable

The BERT-large model has 24 layers. Different layers could have learned different granularities or perspectives of language during the pre-training procedure. For example, Tenney et al. (2019a) observe that most local syntactic phenomena are encoded in lower layers while higher layers capture more complex semantics. They also show that most examples can be classified correctly in the first few layers. From above, we conjecture that lower layers of BERT are more invariant and transferable across tasks.

We divide the layers of the BERT-large model into three groups: low layers (0th-7th layer), middle layers (8th-15th layer), and high layers (16th-23rd layer). As shown in Figure 7, we plot the two-dimensional loss surfaces with respect to different groups of layers (i.e., parameter subspace instead of all parameters) around the fine-tuned point. To be specific, we modify the loss surface function in Section 3.2 to $f(\alpha, \beta) = \mathcal{J}(\boldsymbol{\theta}_1 + \alpha\boldsymbol{\delta}_1^G + \beta\boldsymbol{\delta}_2^G)$, where $\boldsymbol{\theta}_1$ represents the fine-tuned parameters, $G \in \{$low layers, middle layers, high layers$\}$, and the optimization direction of the layer group is used as the axis. On the visualized loss landscapes, $f(0, 0)$ corresponds to the loss value at the fine-tuned point. Besides, $f(-1, 0)$ corresponds to the loss value with the corresponding layer group rollbacked to its original values in the pre-trained BERT model.

Figure 7 shows that the loss surface with respect to lower layers has the wider local optimum along the optimization direction. The results demonstrate that rollbacking parameters of lower layers to their original values (the star-shaped points in Figure 7) does not dramatically hurt the model performance. In contrast, rollbacking high layers makes the model fall into the region with high loss. This phenomenon indicates that the optimization of high layers is critical to fine-tuning whereas lower layers are more invariant and transferable across tasks.

In order to make a further verification, we rollback different layer groups of the fine-tuned model to the parameters of the original pre-trained BERT model. The accuracy results on the development set are presented in Table 1. Similar to Figure 7, the generalization capability does not dramatically decrease after rollbacking low layers or middle layers. Rollbacking low layers (0th-7th layer)

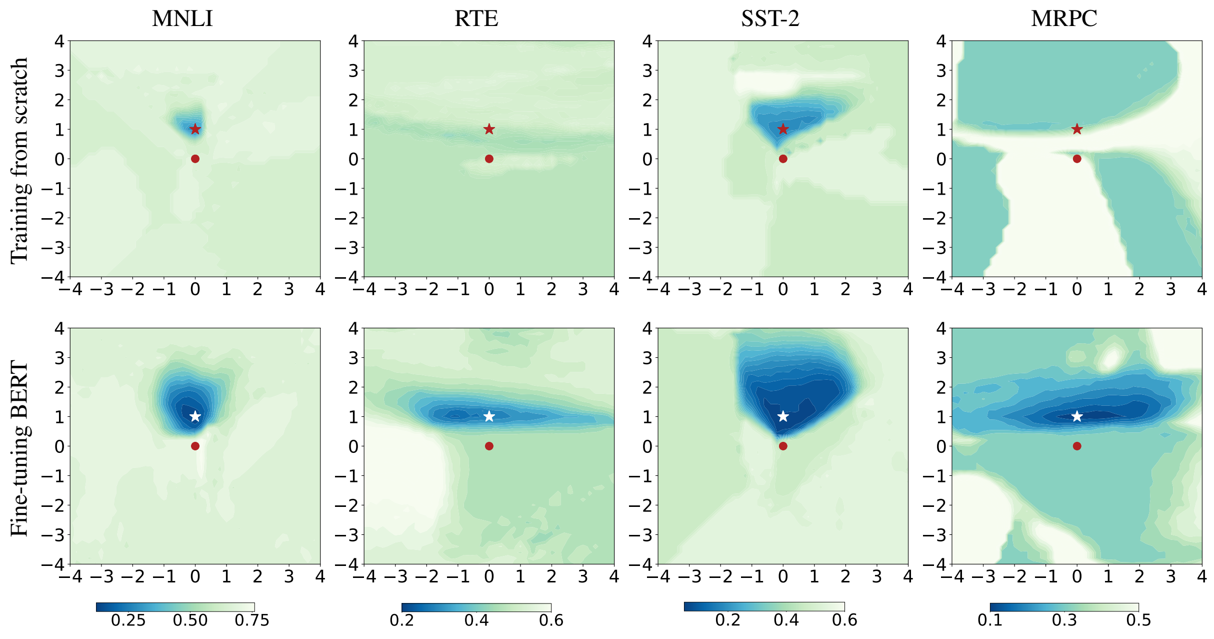

Figure 6: Two-dimensional generalization error surfaces of training from scratch (top) and fine-tuning BERT (bottom). The dot-shaped point and the star-shaped point indicate the generalization errors at the beginning and at the end of the training procedure, respectively.

Dataset	BERT	Layer Rollback		
		0-7	8-15	16-23
MNLI	86.54	86.73 (+0.19)	84.71 (-1.83)	32.85 (-53.88)
RTE	75.45	73.29 (-2.16)	70.04 (-5.41)	47.29 (-28.16)
SST-2	94.04	93.69 (-0.35)	93.12 (-0.92)	59.29 (-34.75)
MRPC	90.20	87.99 (-2.21)	80.15 (-10.05)	78.17 (-12.03)

Table 1: Accuracy on the development sets. The second column represents the fine-tuned BERT models on the specific datasets. The last column represents the fine-tuned models with rollbacking different groups of layers.

even improves the generalization capability on the MNLI dataset. By contrast, rollbacking high layers hurts the model performance. Evaluation results suggest that low layers that are close to input learn more transferable representations of language, which makes them more invariant across tasks. Moreover, high layers seem to play a more important role in learning task-specific information during fine-tuning.

8 Related Work

Pre-trained contextualized word representations learned from language modeling objectives, such as CoVe (McCann et al., 2017), ELMo (Peters et al., 2018), ULMFit (Howard and Ruder, 2018b), GPT (Radford et al., 2018, 2019), and BERT (Devlin et al., 2018), have shown strong performance on a variety of natural language processing tasks.

Recent work of inspecting the effectiveness of the pre-trained models (Linzen et al., 2016; Kuncoro et al., 2018; Tenney et al., 2019b; Liu et al., 2019a) focuses on analyzing the syntactic and semantic properties. Tenney et al. (2019b) and Liu et al. (2019a) suggest that pre-training helps the models to encode much syntactic information and many transferable features through evaluating models on several probing tasks. Goldberg (2019) assesses the syntactic abilities of BERT and draws the similar conclusions. Our work explores the effectiveness of pre-training from another angle. We propose to visualize the loss landscapes and optimization trajectories of the BERT fine-tuning procedure. The visualization results help us to understand the benefits of pre-training in a more intuitive way. More importantly, the geometry of loss landscapes partially explains why fine-tuning

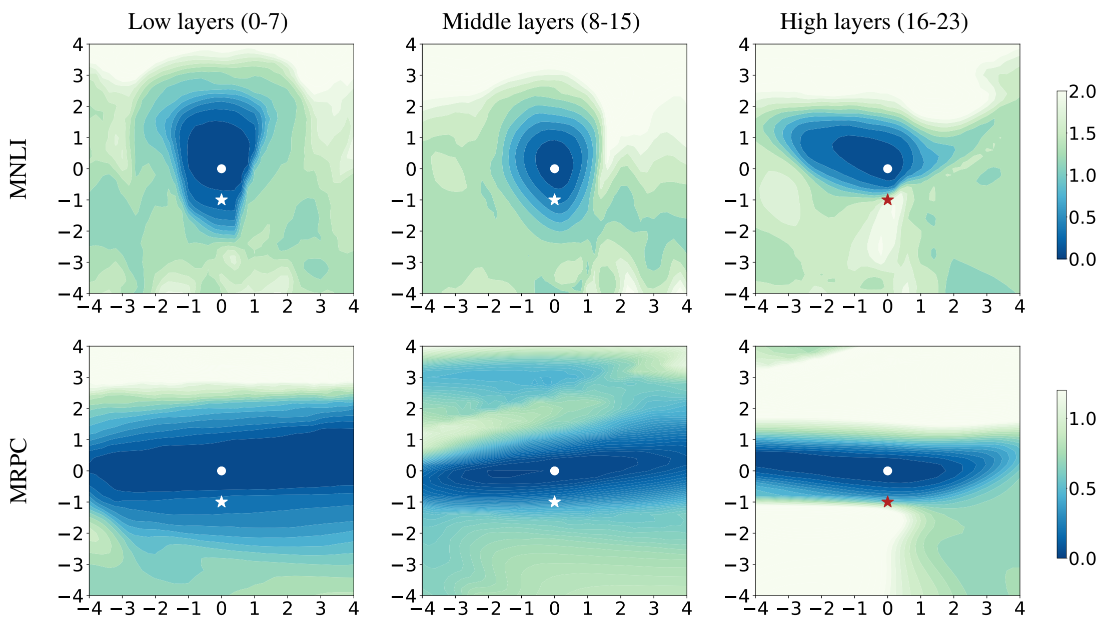

Figure 7: Layer-wise training loss surfaces on the MNLI dataset (top) and the MRPC dataset (bottom). The dot-shaped point represents the fine-tuned model. The star-shaped point represents the model with rollbacking different layer groups. Low layers correspond to the 0th-7th layers of BERT, middle layers correspond to the 8th-15th layers, and high layers correspond to the 16th-23rd layers.

BERT can achieve better generalization capability than training from scratch.

Liu et al. (2019a) find that different layers of BERT exhibit different transferability. Peters et al. (2019) show that the classification tasks build up information mainly in the intermediate and last layers of BERT. Tenney et al. (2019a) observe that low layers of BERT encode more local syntax, while high layers capture more complex semantics. Zhang et al. (2019) also show that not all layers of a deep neural model have equal contributions to model performance. We draw the similar conclusion by visualizing layer-wise loss surface of BERT on downstream tasks. Besides, we find that low layers of BERT are more invariant and transferable across datasets.

In the computer vision community, many efforts have been made to visualize the loss function, and figure out how the geometry of a loss function affects the generalization (Goodfellow and Vinyals, 2015; Im et al., 2016; Li et al., 2018). Hochreiter and Schmidhuber (1997) define the flatness as the size of the connected region around a minimum. Keskar et al. (2016) characterize the definition of flatness using eigenvalues of the Hessian, and conclude that small-batch training converges to flat minima, which leads to good generalization. Li et al. (2018) propose a filter normalization method to reduce the influence of parameter scale, and show that the sharpness of a minimum correlates well with generalization capability. The assumption is also used to design optimization algorithms (Chaudhari et al., 2017; Izmailov et al., 2018), which aims at finding broader optima with better generalization than standard SGD.

9 Conclusion

We visualize the loss landscapes and optimization trajectories of the BERT fine-tuning procedure, which aims at inspecting the effectiveness of language model pre-training. We find that pre-training leads to wider optima on the loss landscape, and eases optimization compared with training from scratch. Moreover, we give evidence that the pre-training-then-fine-tuning paradigm is robust to overfitting. We also demonstrate the consistency between the training loss surfaces and the generalization error surfaces, which explains why pre-training improves the generalization capability. In addition, we find that low layers of the BERT model are more invariant and transferable across tasks.

All our experiments and conclusions were derived from BERT fine-tuning. A further understanding of how multi-task training with BERT (Liu et al., 2019b) improves fine-tuning, and how it affects the geometry of loss surfaces are worthy of exploration, which we leave to future work. Moreover, the results motivate us to develop fine-tuning algorithms that converge to wider and more flat optima, which would lead to better generalization on unseen data. In addition, we would like to apply the proposed methods for other pre-trained models.

Acknowledgements

The work was partially supported by National Natural Science Foundation of China (NSFC) [Grant No. 61421003].

References

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. *arXiv preprint arXiv:1903.07785*.

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

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC09*.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. 2017. Entropy-SGD: Biasing gradient descent into wide valleys.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. 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, pages 177–190, Berlin, Heidelberg. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

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, Prague. Association for Computational Linguistics.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287.

Ian J. Goodfellow and Oriol Vinyals. 2015. Qualitatively characterizing neural network optimization problems. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Flat minima. *Neural Comput.*, 9(1):1–42.

Jeremy Howard and Sebastian Ruder. 2018a. 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.

Jeremy Howard and Sebastian Ruder. 2018b. Universal language model fine-tuning for text classification. In *ACL*. Association for Computational Linguistics.

Daniel Jiwoong Im, Michael Tao, and Kristin Branson. 2016. An empirical analysis of deep network loss surfaces. *CoRR*, abs/1612.04010.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, San Diego, CA.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *CoRR*, abs/1611.01368.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. *CoRR*, abs/1903.08855.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? Adapting pretrained representations to diverse tasks.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? Probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

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. In *International Conference on Learning Representations*.

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, New Orleans, Louisiana. Association for Computational Linguistics.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. 2019. Are all layers created equal? *arXiv preprint arXiv:1902.01996*.