How much does pre-trained information help? Partially re-initializing BERT during fine-tuning to analyze the contribution of layers

Stanford CS224N {Custom} Project

Trisha Singh
Statistics
Stanford University
trsingh@stanford.edu

Davide Giovanardi
ICME
Stanford University
davide3@stanford.edu

Abstract

Transformer models like BERT are used extensively in NLP tasks. We add to the growing literature on investigating how BERT works by analyzing the contribution of pre-trained knowledge for each layer on downstream tasks. We introduce a method called partial re-initialization to understand the performance gains for each layer from starting with pre-trained parameters and then fine-tuning versus starting from randomly initialized parameters. We compare this performance across three GLUE tasks (SST, QNLI, and CoLA) and test how this performance varies by training data size. We also study how the weights for each layer change from the pre-trained BERT model during fine-tuning. In general, we see that when there is less training data, there is more reliance on pre-trained weights, which leads to large performance gains over a fully supervised model with more training data. As training data size increases, the performance gains from the pre-trained information of each layer are lower, subject to the nature of the task. We also find that the parameters of the later layers get overwritten more during fine-tuning, while the earlier layers remain largely invariant. With more training data, the parameters are transformed more.

1 Key Information

- Mentor: Alex Tamkin

2 Introduction

Pre-trained contextual word embeddings are widely used in many NLP tasks, and tend to provide large performance gains. These embeddings are typically trained on unlabelled data with the goal of learning a general representation of the word or input token. With contextual models like transformers, each token is assigned an embedding that encodes information from the entire input sequence. This is in contrast to static representations in which each token is assigned a fixed embedding. Further, in BERT, this encoding is bi-directional (thus mitigating dependence on the order of the sentence) and the deep nature of the model allows for modelling long-range dependencies [1]. The pre-trained information encoded in the embeddings provided by BERT are shown to greatly improve performance in downstream tasks.

The success of models like BERT on a wide range of NLP tasks suggests that its layers encode some features of language that are generalizable and may correspond to different types of linguistic knowledge. There has been a lot of innovative work on studying the type of knowledge that BERT encodes and where it is encoded. One such method is probing, which aims to understand the nature of the information encoded by different layers in the model. In general, the approach is to obtain a token

representation from the layer being probed and to use it as fixed input to a simple probing classifier. These strategies have helped answer questions about how the layers of BERT map to steps in the NLP pipeline, the hierarchical nature of these layers, the extent of contextual information encoded in its during pre-training, among many others.

But the question still remains, what happens when the model is actually fine-tuned? We build upon the knowledge about the layers of BERT and extend it to ask the question: How much does the pre-trained knowledge in each layer contribute to downstream performance, depending on the task and the size of training data? This aims to provide a different and more nuanced perspective on how the contextual pre-trained information from BERT is useful.

We introduce a procedure which we will call partial reinitialization. We load the pre-trained information for the layers being examined, randomly reinitialize the parameters of the other layers, and then fine-tune this model on the downstream task. We allow the weights of layers being examined to update during training instead of treating them as fixed representations.

Our main contributions are:

- We see clear patterns in how training data size affects learning during fine-tuning. Pre-trained information from less layers is used as more data becomes available. On the other hand, we see large performance gains from having pre-trained information in small data settings.
- We provide insight in the distinction between using existing information (probing) versus how easily pre-trained information can be transformed (partial re-initialization). The embeddings from BERT's layers are transferable not as-is but with some supervised help. In probing, we see that pre-trained information from more layers is needed than in partial reinitialization to achieve comparable downstream perform.
- We observe how the model weights for each layer change during fine-tuning, and find that it is sensitive to training data size. With more training data, the weights for all the layers are transformed more. The earlier layers remain largely invariant while the later layers change more.

We introduce our partial reinitialization method, conduct experiments while varying training data size, perform probing experiments for comparison, and also analyze how parameters change during fine-tuning.

3 Related Work

3.1 Probing

There have been many studies to analyze the type of information captured by each layer of BERT through probing. This is done by designing specific tasks, which are simpler than downstream tasks, that map to some component of the NLP pipeline which requires understanding of varying levels of syntactic and semantic information. It has been shown that the pre-trained layers of BERT hierarchically encode syntactic and semantic information [2, 3]. Attempts to study where such information is encoded indicate that that the lower layers encode lower-level syntactic structure and as more layers are added, the model captures more complex relationships [4, 5].

There has also been work on understanding how these pre-trained representations contribute to downstream task performance as fixed inputs. Tenney et al. (2019) find that the pre-trained knowledge from BERT provides larger gains on syntactic tasks and smaller gains on semantic ones [6]. There is also a wide consensus that the middle layers of BERT encode the most syntactic information and that the final layers are the most task-specific [5, 7].

3.2 Other methods

Liu at al. (2019) examine the transferability of contextual word representations and find that the task-specific nature of later layers in transformers is not monotone, which also motivates our analysis of the contribution of each layer with the partial reinitialization perspective [2]. Hao et al. (2019) examine the gradient descent path during fine-tuning with pre-trained weights and compare it to a

model learned from scratch. They find that starting from a pre-trained model leads to wider optima and that fine-tuning is robust to overfitting [8]. We analyze if this translates into performance gains from pre-trained knowledge in small data settings.

4 Approach

4.1 Introduction

Our overarching goal is to provide insights on why the pre-trained information from BERT works well on downstream tasks by building upon past research about the information encoded by each layer of BERT. We test this by starting from a pre-trained BERT model and fine-tuning it. At each stage in the experiment, we incrementally re-initialize an additional layer so that its information from pre-training is lost and it is trained from scratch during fine-tuning. In particular, we check for the following:

- We examine the **differential effect of layers within tasks** to see if there are minimal performance losses from partial re-initialization of the last few layers as compared to the earlier and middle layers, which are supposed to contain more transferable knowledge.
- We test the **differential effect of layers across tasks** to see how the performance due to partial reinitialization of each additional depends on the nature of the task.
- We also investigate the effect of **data size** on the partial re-initialization strategy to test if the pre-trained weights acquire more importance as less training data is available, and how this varies by layer.
- Finally, we analyze how the output layers' **weights change** between pre-trained models with and without fine-tuning.

4.2 Baseline

We use the BERT-base model which contains 12 layers. Each layer is comprised of three sub-layers: (1) Self-Attention, (2) Intermediate, and (3) Output. Overall, there are 6 linear projections performed within each of the 12 BERT layers. Given a downstream task, the most common way to use the BERT model is to load it with pre-trained weights and then train this model on the specific task. This is known as *fine-tuning* (Fig 4).

4.3 Partial re-initialization

Our approach focuses on the dynamic nature of training a model and studies the role of pre-trained knowledge when all layers' parameters are allowed to change and interact with one-another during fine-tuning. We try to answer the following questions: To what extent pre-trained information (layers' weights) helps fine-tuning to achieve high accuracy for a downstream task? Is the contribution different across layers?

Our procedure is as follows:

- We use the Transformers framework from huggingface (`https://github.com/huggingface/transformers`). This provides us with the methods to load a pre-trained BERT model and a script to perform fine-tuning. We modify a script in this library to download SST-2, QNLI, and CoLA data from the GLUE benchmarks tasks.
- We modify the fine-tuning process by adding a method that re-initializes the layers of BERT using the truncated normal distribution. In the original BERT release source code the authors use a truncated normal density with mean μ zero and standard deviation σ equal to 0.02 to initialize layers; furthermore, the tensorflow documentation specifies that we truncate according to a range $[-2\sigma, 2\sigma]$. We replicate this method in PyTorch.
- For each selected layer, we re-initialize all six linear projections that are distributed as follows: – BERT Self-Attention (3): Key, Query, Value – BERT Self-Output (1) – BERT Intermediate (1) – BERT Output (1)
- We create a wrapper script in bash that specifies the desired parameters and run the 12 experiments for a given fine-tuning task with specified training data size.

Algorithm 1 Partial re-initialization

procedure PARTIALREINIT($M, layers\_to\_init, \sigma$)
  for $layer$ in M **do**
    if $layer$ in $layers\_to\_init$ **then**
      for $weight$ in $layer$ **do** ▷ We re-init all linear projections
        $weight \leftarrow trunc\_norm(\mu = 0, \sigma)$ ▷ Truncates at $\pm 2\sigma$

4.3.1 Dataset size

We look into how the size of the data can have a influence on our method. Intuitively, the less the data-points available, the more relevant the pre-trained information carried by each layer. We focus on two downstream tasks, QNLI and SST-2 and investigate three different sizes: $100\%, 10\%, 1\%$.

Moreover, we analyze a direct comparison between the two tasks and study how a low-data scenario affects the performance, layer by layer. To make the size equal for both data-sets, we run the experiment on 1% and 0.6% for SST-2 and QNLI respectively.

4.4 Probing

We also conduct probing on the same downstream tasks used in partial re-initialization. This method allows us to look at pre-trained information as fixed inputs and thus we seek to compare the results with our approach, in which generalizability is more loosely defined as the ability to usefully transform pre-trained information with supervision. In probing, the goal is to understand the contribution of pre-trained information of each layer, conditional on the information carried by the preceding layers. To achieve this, we progressively discard k layers, for $k \in [1, \ldots, 11]$, freeze the remaining layers' parameters, and train a classifier to make predictions. Because all the layers' parameters are frozen, the training will only involve a classifier. We provide a short pseudo-code that illustrates how we modify the model before beginning train the classifier and make predictions on the test set.

Algorithm 2 Probing

procedure PROBING($M, layers\_to\_remove$)
  for $layer$ in M **do**
    if $layer$ in $layers\_to\_remove$ **then**
      Remove layer
    else ▷ Freeze all parameters for layers kept
      for $param$ in $layer$ **do**
        Freeze

4.5 Weight similarity

We carry out a weight similarity analysis on a layer-by-layer basis between the *bert-pretrained* and *bert-finetuned* models. To achieve this, we compute the euclidean distance between the weight matrices of the linear projections contained in the output sub-layer.

Moreover, we implement this analysis for the three dataset sizes mentioned in section 4.3.1.

5 Experiments

5.1 Data

We use data from the GLUE benchmark used for testing the performance of models on natural language understanding tasks [9]. There are nine tasks, of which we choose: (1) The Stanford Sentiment Treebank (SST-2), (2) Question-answering Natural Language Understanding (QNLI), and (3) The Corpus of Linguistic Acceptability (CoLA). All these tasks are single sentence or sentence-pair classification tasks, which gives us a uniform way to compare performance on downstream tasks with different linguistic properties.

Table 1: Task description and statstics. SST-2 and CoLA are single sentence classification tasks, while QNLI is a sentence-pair classification task.

Task	|Train|	|Test|	Input and label (input, label)	Evaluation metric
SST-2	67k	1.8k	(sentence, positive/negative)	Accuracy
QNLI	105k	5.4k	((question, paragraph), answers question yes/no)	Accuracy
CoLA	8.5k	1k	(sentence, acceptable/not acceptable)	MCC

We choose these three tasks in particular because SST and QNLI offer variation in type of task, while having a large enough dataset size to be able to subset in order to test our hypothesis about reliance on training data size. Although CoLA is small, we wanted to see how BERT would do on a task aimed at evaluating grammatical accuracy, a task which does not correlate with BERT pre-training tasks. We provide a short description of the tasks, and further details are provided in Table 1.

- **SST-2:** These examples are obtained from the Stanford Sentiment Treebank corpus [10]. The input sentences are movie reviews and the label is a positive or negative sentiment classification.
- **QNLI:** This is formed from the Stanford Question Answering Dataset [11]. The input is question-paragraph pair and the label classifies whether the paragraph (drawn from Wikipedia) contains the answer to the question.
- **CoLA:** This task is developed from the Corpus od Linguistic Acceptability [12]. The input is a sequence of words and the task is to label whether this sequence is an acceptable sentence by English grammatical standards.

5.2 Evaluation method

To evaluate the performance of our method, we compute *accuracy* for SST-2 and QNLI and *Matthews Correlation Coefficient* for CoLA. We compute these metrics always on a test set which is never seen by the model during training.
Accuracy measures the ratio of correctly predicted labels over the size of the test set. Formally:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Since CoLA presents class imbalances, we use the MCC which is better suited for unbalanced binary classifiers [12]. It measures the correlation of two Boolean distributions, giving a value between -1 and 1. A value of 0 means that the two distributions are uncorrelated, regardless of any class imbalance.

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN))}{\sqrt{(TP + FP)(TP + FN)(FP + TN)(TN + FN)}}$$

Finally, we implement L2 distance to measure the similarity of output-layer weights between different models:

$$d(W_{base}, W_{FT}) = \|W_{base} - W_{FT}\|_2$$

where W_{base} and W_{FT} are the weights matrices from *bert-base* and *bert-fine-tuned* respectively.

5.3 Experimental details

• Learning rate: 2e-5 • Epochs: 3 • Iterations/epoch: 2000

• Time per experiment: 2h 40min for QNLI, 2h 20min for SST-2, 20min for CoLA

6 Results

6.1 Partial re-initialization

6.1.1 SST-2

From Fig 1 (a), we see that the performance gains from pre-trained information for different layers vary a lot by training data size. The loss in accuracy with increasing partial re-initialization becomes less concave as training data size decreases. This signifies that for this task, the contribution of pre-trained information from the middle layers is much higher with smaller training data size.

Another interesting thing to note is that for 1% of the training examples (around 670 examples), the accuracy of a model with no or partial re-initialization of the last few layers is actually higher than that of a fully supervised model trained on 10 or 100 times the training data.

6.1.2 QNLI

From Fig 1(b), we see that QNLI displays a different trend. For all the training data fractions, the contribution of pre-trained information from the middle layers is fairly low, as seen by the concavity of the drop in accuracy until partial re-initialization hits the earlier layers. An interesting thing to note is that the drop in accuracy from missing the contextual information from the last few layers is shifted more towards earlier layers as training data size increases. In other words, in the case of full training data, the accuracy drops rapidly after partial reinitialization begins at the first three layers, in the 10% training data case, the accuracy drops most for the first four layers, and in the 1% training data case, the rapid drop begins at the first five layers. Thus, as training data size decreases, the pre-trained information from more and more earlier layers becomes important.

6.1.3 SST-2 vs QNLI for small data

In the case of both SST-2 and QNLI, we see that having pre-trained information from BERT leads to fairly large performance gains, even though the contribution of different layers varies across these tasks. This is illustrated in Fig 1(c). In order to see how the importance of BERT's pre-trained information varies by the nature of the task, we compare the fine-tuning performance with partial reinitialization on both tasks with similar training data sizes (around 670 examples). We see very different degrees of concavity. For QNLI, the accuracy drop is concave, with the biggest drop for the last four layers, with reinitialization of the last three layers leading to accuracy equal to the null error rate. Thus, the pre-trained information from the first four layers contributes most, with the middle and later layers not contributing much. For SST-2, we see that reinitializing the middle layers also leads to considerable drops in accuracy, by about 5% points. As expected, the earlier layers also lead to large drops. This matches with the intuition that SST-2 relies on more semantic information and thus uses the information that is found to be encoded in BERT's middle layers, while QNLI relies more on syntactic information and word similarity, and thus masks the most use of the earlier lower-level word representations encoded by BERT's earlier layers.

6.1.4 CoLA

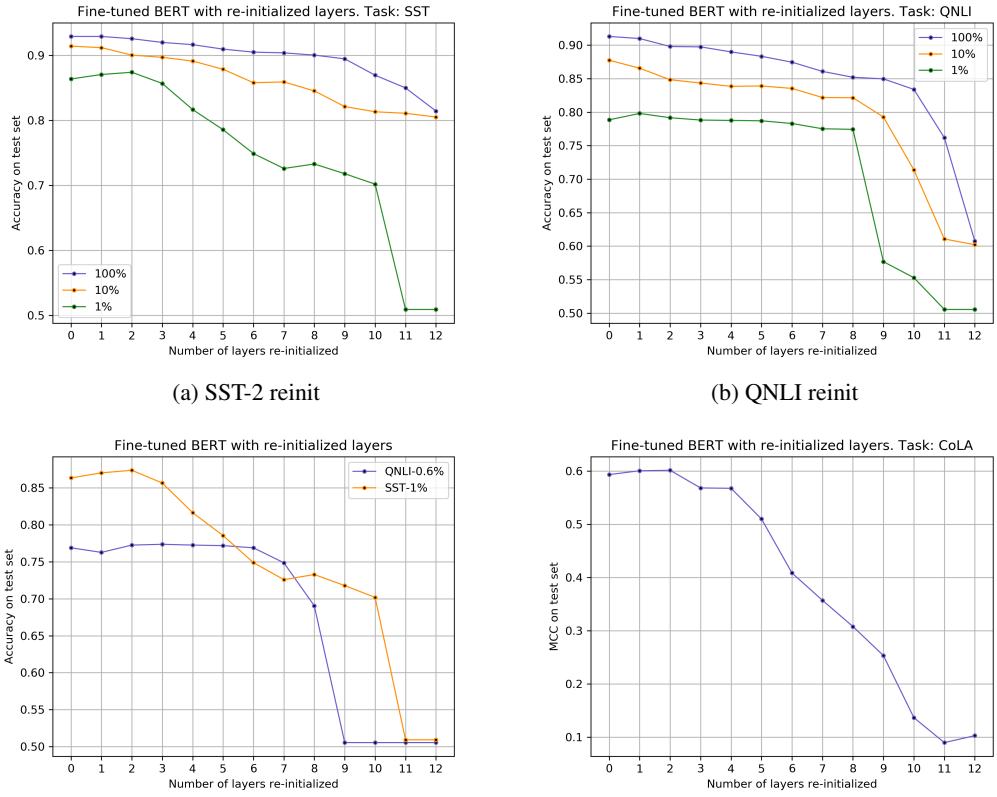
As seen in Fig 1 (d), we see surprising results for CoLA, with a fairly linear drop in performance with increasing partial reinitialization. Thus, the pre-trained information from middle and earlier layers seems to contribute equally to improving the correlation as measured by MCC. A possible explanation is that the data size for CoLA is fairly small (around 8500 examples), the pre-trained weights for each layer provide a good starting point for the weights to be transformed in task-specific manner.

6.2 Probing

We focus on SST-2 and QNLI downstream tasks, and refer to Fig 2 for the following analysis[1].

Probing reaches a lower peak accuracy than partial re-initialization; this result is expected as the fine-tuning is limited to only training a classifier on top of k frozen layers. An important difference

[1]Results for probing on CoLA can be found in the Appendix.



(a) SST-2 reinit

(b) QNLI reinit

(c) SST vs QNLI small data reinit

(d) CoLA reinit

Figure 1: **Partial re-initialization results**

with partial re-initialization is that in probing the model needs more layers to enable a considerable improvement in performance. We can visualize this finding in Fig 2 for both SST-2 and QNLI: we need to remove less layers in order to see the jump in accuracy.

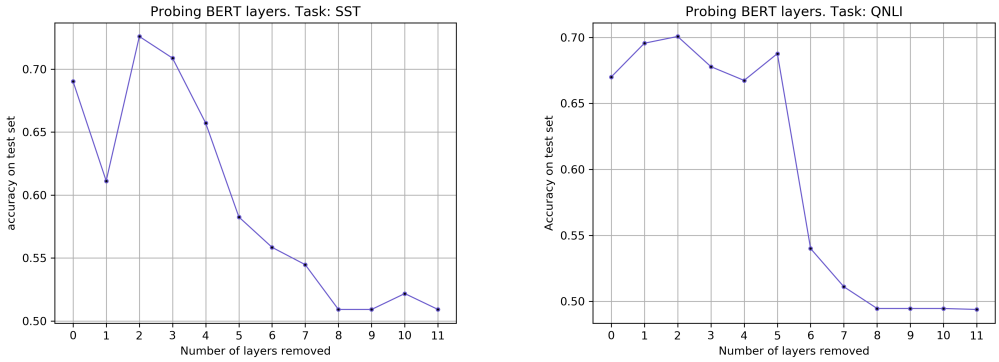
Another important difference with partial re-initialization is that the highest two layers hurt the performance in both SST-2 and QNLI, while in the re-initialization method they just have a negligible effect on accuracy, i.e. they don't bring any new information. The behavior seen in probing can be explained by the fact that the highest two layers are the most task-specific and thus their parameters are the most overwritten during fine-tuning; since we freeze all layers' parameters in probing, it's likely that the pre-trained information is inappropriate for the two downstream tasks and results in hurting the performance.

Finally, if we compare the probing results between the two tasks, we can see that they present significant differences in their patterns: the accuracy on SST-2 decreases more gradually than QNLI, which instead presents a steep decline when we remove the the sixth layer. This divergence matches with the partial re-initialization results [Fig 1 (a, b)] where SST-2 presents a more linear pattern than QNLI which instead presents a steep jump in performance.

6.3 Weight similarity

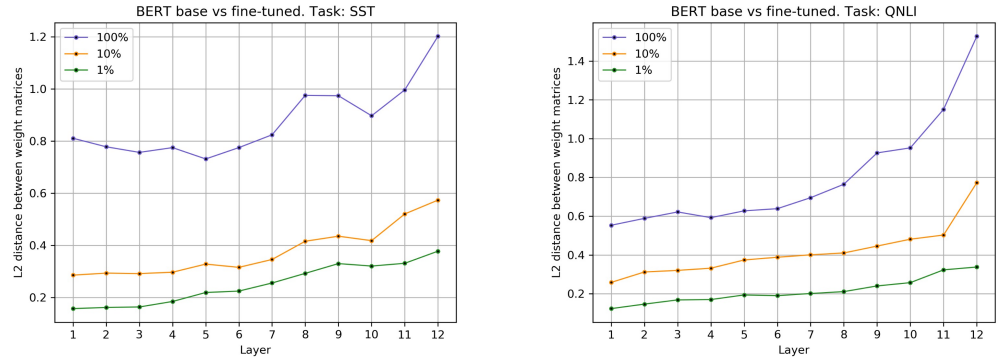
We plot the results for the weight similarity analysis in Fig 3.

We notice parallel upward shifts with increased data size. Intuitively, with more data available, training can exploit a larger amount of information; as a consequence, all weights are likely to experience a bigger change in magnitude compared to the less data scenarios.



(a) SST-2 probing (b) QNLI probing

Figure 2: **Probing experiments**



(a) SST-2 parameter similarity (b) QNLI parameter similarity

Figure 3: **Parameter similarity**. We compare between pre-trained BERT base and fine-tuned BERT base measured by L2 distance between weight matrices

We see that in all experiments, the curves are upward trending, indicating that higher layers experience a bigger change compared to lower layers. This is yet another confirmation of the intuition that higher layers are the most task-specific and are thus expected to undergo bigger changes.

Finally, we observe that the increase in L2 distance is steeper when the data size is bigger. As mentioned above, more data means more information to update pre-existing knowledge with; as a consequence, because the higher layers are the most task-specific, it seems reasonable to assume that they are the ones to use this additional information the most.

7 Conclusion

In this paper, we developed a novel method, partial re-initialization, to analyze the contribution of BERT's pre-trained information in the context of fine-tuning. We explored the influence of data and found that pre-training helps when data is scarce. Moreover, we applied our method to different tasks and discovered that the importance of middle-layers' pre-trained information is task dependant: while it increases performance in SST-2, it fails to provide a positive contribution in QNLI.

In an effort to show the mechanics of our dynamic approach, we compared partial re-initialization to probing, where all layers' parameters are frozen; we showed that with partial re-initialization, the model needs less pre-trained information in order to exhibit a significant increase in performance.

Finally, we looked for patterns in how weights change between the base and fine-tuned model and saw that the weights' changes more with more data, and earlier layers generally remain invariant.

References

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

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

[3] John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[4] Yongjie Lin, Yi Chern Tan, and Robert Frank. Open sesame: Getting inside bert's linguistic knowledge. *CoRR*, abs/1906.01698, 2019.

[5] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *arXiv preprint arXiv:2002.12327*, 2020.

[6] 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. What do you learn from context? probing for sentence structure in contextualized word representations. *CoRR*, abs/1905.06316, 2019.

[7] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets 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 4365–4374, Hong Kong, China, November 2019. Association for Computational Linguistics.

[8] Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Visualizing and understanding the effectiveness of bert. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[9] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018.

[10] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 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, October 2013. Association for Computational Linguistics.

[11] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[12] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *CoRR*, abs/1805.12471, 2018.

A Appendix A: Additional Figures

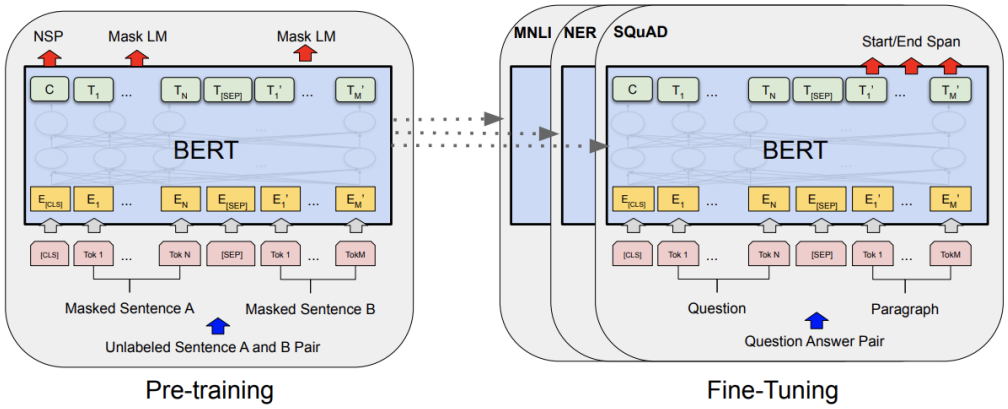
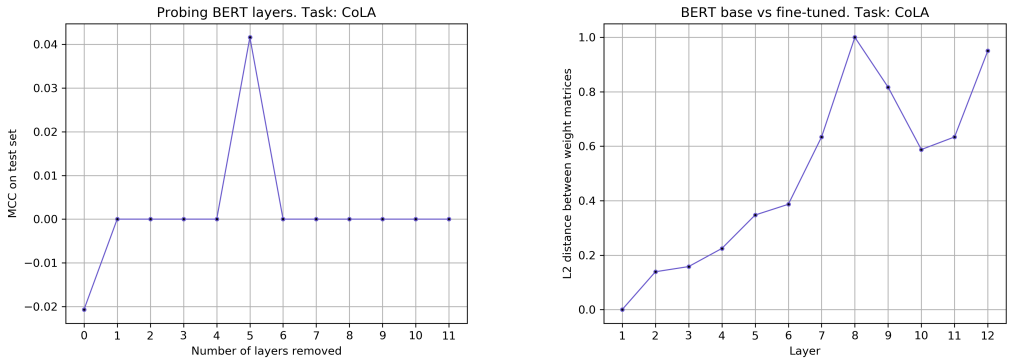


Figure 4: **Fine-tuning with BERT** (from Devlin et al., 2018 [1])

B Appendix B: CoLA

We discuss the probing and weight similarity results for the CoLA downstream task. We draw the following observations:

- Since BERT base is only trained on masked language modelling (MLM) and next sentence prediction (NSP), its layers are not expected to learn knowledge about grammatical structure. We can clearly see this from Fig 5 (a) where the MCC is zero for all but two data-points (an MCC of 0 corresponds to uninformed guessing in a classification task).
- Regarding the weights' changes between BERT base and fine-tuned, we see in Fig 5 (b) that for CoLA we have a more noisy pattern. Leveraging the previous discussion that the information encoded in BERT-base is not suited for CoLA, we can conclude that all layers have a more balanced chance to change because none of them is appropriate for this specific task.



(a) CoLA probing

(b) CoLA parameter similarity

Figure 5: **Additional results for CoLA**