BERT Goes to College: Understanding the Role of Pretrained Layers During Finetuning

Stanford CS224N Custom Project

Daniel Huang
Department of Computer Science
Stanford University
dhuang7@stanford.edu

Jacky Lin
Department of Computer Science
Stanford University
jackylin@stanford.edu

Shan Zhou
Department of Computer Science
Stanford University
shanzhou@stanford.edu

Abstract

We investigate the extent to which pretraining contributes to BERT's success in producing state-of-the-art results, and identify some modifications to BERT's pretraining process that outperform the existing configuration. We conduct three experiments at varying levels of granularity. First, we sequentially finetune on multiple tasks and find cases where finetuning BERT on additional intermediate tasks actually yields better performance on the final finetuning task, compared to directly finetuning the pretrained layers on the final task. Secondly, within a specific finetuning task, we reinitialize various pretrained layers of BERT and observe the performance across different layerwise reinitialization schemes as well as across different tasks, and compare with the baseline model with no layers reinitialized. We find that reinitializing downstream layers causes a smaller decrease in performance than reinitializing upstream layers, but the change in performance does not monotonically decrease when reinitializing the next upstream layer, nor is the change in performance constant. This result indicates the presence of interlayer dependencies and unequal importance in the information encoded within each layer. Third, we reinitialize various attention heads within each layer and find that reinitializing certain attention heads at top layers can even improve the model performance. Altogether, this project lends insight to the contribution of pretraining to BERT's performance on finetuning tasks, and proposes various ways to modify the pretraining process to further increase performance.

1 Key Information

- Mentor: John Hewitt (CS224N Teaching Assistant) and Alex Tamkin (Stanford NLP Group)
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

The BERT model for language representation produces state-of-the-art results by finetuning the pre-trained BERT model on a specific task. Our project focuses on the broad question of how transfer learning contributes to BERT's performance. We investigate how prior learning (the pretrained layers of BERT) affects future learning (finetuning on a specific task). We approach this problem in varying levels of granularity with mulitple experiments. On the task-level, we sequentially finetune on multiple tasks. On the layer level, we reinitialize different layers. And within each layer, we also modify individual attention heads.

3 Related Work

For each of our experiments, we form our hypotheses based off of results obtained from previous research papers.

"BERT Rediscovers the Classical NLP Pipeline": Tenney et al. [2018] find that BERT encodes the steps of an NLP pipeline sequentially. The authors use different edge probing tasks to investigate the informational content encoded in each pretrained layer. Our experiment extends this research by tracking the performance based on layer-by-layer reinitializations of the BERT pretrained model layers on finetuning tasks. Rather then focusing on the specific information encoded within a layer, we instead investigate how the presence of pretrained layers affect the final model performance after finetuning.

"Are Sixteen Heads Really Better than One?": Michel et al. [2019] find that test performance does not significantly decrease if attention heads are removed after training. Our experiment is similar to this research in that we reinitialize different attention heads in different layers before finetuning.

"Taskonomy": Zamir et al. [2018] determine the relationships between various computer vision tasks, and uses transfer learning to reduce the amount of data needed to train on a group of related tasks. Our experiment investigates the similarity between different finetuning tasks by sequentially finetuning on multiple tasks. In certain cases, we find that finetuning on an additional intermediate task before the final task actually produces superior performance compared to directly finetuning on the final task.

We also drew inspiration from the in-class lecture by Jacob Devlin from Google AI. Google, as a search-engine company is motivated in delivering timely results to search queries. As such, it seeks to use NLP models such as BERT to improve results returned to user. However, given billions of searchs that need to be returned in milliseconds, BERT base and BERT large are too computationally intensive to be used for search queries. Instead Devlin and his team have distilled the BERT model for specific finetuned task.

Devlin and his team found that distilling the BERT model and then finetuning leads to very poor results compared to the original BERT model that has been finetuned. However, finetuning the original BERT model and then distilling the model leads to comparable performance compared to the control (the original BERT model that has been finetuned for a specific task).

4 Approach

We approach this problem using multiple experiments to change the model and investigate the resulting change in performance. In experiment one, we sequentially finetune on multiple tasks. In experiment two, we reinitialized different layers. In experiment three, we reinitialized certain attention heads.

4.1 Experiment 1

4.1.1 Description

We sequentially finetune on multiple tasks. Specifically, we begin with BERT's pretrained layers. Then, we first finetune on an intermediate task (task 1). We then use this resulting model and finetune on the final task (task 2). For each final task, we run experiments with different intermediate tasks, and compare the results to the baseline (finetune directly on the final task without an intermediate task).

4.1.2 Hypothesis

We expect a higher performance when finetuning on an intermediate task and final task that are similar to each other, compared to finetuning on two tasks that are dissimilar. For instance, given tasks A, B, and C, where task A is similar to task B but dissimilar to task C, then finetuning on B then A will yield a higher performance than finetuning on C then A. We also do not expect that finetuning on two tasks will be able to beat the performance of directly finetuning on the final task (without an intermediate task), since presumably BERT's pretrained layers should already contain sufficient linguistic information.

4.2 Experiment 2

4.2.1 Description

We reinitialize the last $k \in [1, 12]$ pretrained layers of the BERT model and then run the finetuning task. For instance, if $k = 4$, then we use a BERT model, where layers 1 to 3 have the pretrained weights, and layers 4 to 12 have reinitialized weights. We choose to reinitialize all layers that are greater than equal to k instead of just reinitializing layer k, since some downstream layers have dependencies on upstream layers, which would be a confounding variable that affects the model performance.

4.2.2 Hypothesis

We hypothesize that as k increases, performance on the finetuning task will also increase. More fundamental language knowledge, such as part of speech tagging and parsing, is encoded in the upsteam layers, so reinitializing those layers will cause a steeper drop in performance during testing. Since finetuning takes fewer epochs than pretraining, we would not expect the model to be able to relearn these fundamental semantic and syntactic structures in only a few epochs during finetuning.

4.3 Experiment 3

4.3.1 Description

There are 12 attention heads per layer, and 12 total layers in BERT base model. For two different GLUE tasks, we separately reinitialize each attention head at a time for layers 1, 2, 11, and 12. In total, we perform 96 runs (12 attention heads * 4 layers * 2 tasks).

4.3.2 Hypothesis

We hypothesize that reinitializing attention heads will not significantly impact performance, i.e. we do not expect performance to scale linearly with the number of attention heads. We also reinitialize attention heads in upstream and downstream layers. We expect that reinitializing attention heads in downstream layers will cause a smaller decline in performance compared to reinitializing attention heads in upstream layers. Our logic is similar to that of Experiment 2's hypothesis.

4.4 Reinitialization Procedure

We reinitialize all pre-trained layers except the embeddings using the same initialization scheme as the original BERT pretrained layers, which we found in BERT's configuration file. Specifically, we used the truncated normal function with mean of 0 and standard deviation of 0.2. Formally, its probability density function is as follows

$$f(x;\mu,\sigma,a,b) = \frac{1}{\sigma}\frac{\Phi(\frac{x-\mu}{\sigma})}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})}$$

Where the probability density function of the standard normal distribution is defined as: $\Phi(\xi) = \frac{1}{\sqrt{2\pi}} exp(-\frac{1}{2}\xi^2)$

And the cumulative distribution function of the standard normal distribution is defined as: $\Phi(x) = \frac{1}{2}(1 + erf(\frac{x}{\sqrt{2}}))$

For the BERT model initialization, the bounds are $a = -0.04$ and $b = 0.04$, with mean $\mu = 0$ and standard deviation $\sigma = 0.02$, the values used by Devlin et al. [2018].

5 Experiments

5.1 Finetuning Tasks and Data

5.1.1 Question Answering with SQuAD

For the question-answering task, we use SQuAD 2.0, which draws its content from Wikipedia articles. For a question, there are possible answers, which are spans from the corresponding passage on Wikipedia. In SQuAD 2.0, there are also questions with no answers contained in the passage. There are 150,000 questions in the dataset. The evaluation metrics used are F1 score and EM score.

5.1.2 Paraphrase identification with MRPC

We use the Microsoft Research Paraphrase Corpus (MRPC) for paraphrase identification. This task involves classifying if a pair of sentences are paraphrases or not. There are 5,800 sentence pairs in the dataset. The evaluation metrics used are F1 score and accuracy.

5.1.3 Recognizing Textual Entailment (RTE)

This dataset gives pairs of text fragments and the task is to recognize if one of the fragment can be logically entailed from the other fragment. There are 2,500 examples in the dataset. The evaluation metric used is accuracy.

5.1.4 Corpus of Linguistic Acceptability (CoLA)

The CoLA task evaluates if a sentence is grammatical/acceptable. The dataset contains 10,657 annotated sentences from 23 linguistics publications. The evaluation metric used is Matthew's Correlation Coefficient.

5.1.5 Winograd Natural Language Inference (WNLI)

WNLI is a natural language inference task whereby two sentences differ by one or two words, which creates an ambiguity that can be resolved by reasoning. The dataset contains 150 schemas. The evaluation metric used is accuracy.

5.1.6 Semantic Textual Similarity Benchmark (STS-B)

STS-B is a task that measures the similarity of sentence representations based on their cosine similarities. The dataset contains 8,628 sentence pairs. The evaluation metric used is Pearson/Spearman correlations.

5.2 Evaluation method

$$\text{precision} = \frac{\text{number of matched tokens}}{\text{number of predicted tokens}}$$

$$\text{recall} = \frac{\text{number of matched tokens}}{\text{number of ground truth tokens}}$$

From the precision and recall, we obtain the F1 score:

$$F_1 = \left(\frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}\right) = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

We will also use the exact match (EM) score. This is calculated as:

$$\text{EM} = \frac{\text{Number of exact matches}}{\text{Number of test examples}}$$

$$\text{accuracy} = \frac{\text{number of matched tokens}}{\text{total number of tokens}}$$

Correlation metrics:

The Matthew's correlation coefficient ranges from -1 to 1, used as an evaluation metric for unbalanced binary classifiers.

$$\text{mcc} = \frac{\text{True Positive * True Negative - False Positive * False Negative}}{\text{True Positive + False Positive * False Negative + False Positive + True Negative * True Positive + False Negative}^{1/2}}$$

The Pearson and Spearman correlation coefficients both range from -1 to +1. The Pearson correlation metric depicts linear relationships while the Spearman correlation metric depicts monotonic relationships.

$$\rho_p = \frac{\text{cov}(X,Y)}{\sigma_x \sigma_y}$$

$$\rho_s = 1 - \frac{6 \sum d_i^2}{n(n^2-1)}$$

5.3 Experimental details

For all experiments, we used the BERT base cased model. Our learning rate was $2 * 10^{-5}$, and batch size was 1 for SQuAD (due to limited CUDA memory). For GLUE tasks, we used a batch size of 32.

5.3.1 Experiment 1: finetuning on multiple tasks

We finetune with 3 epochs on the first finetuning task, and then 3 epochs on the second finetuning task. We ensure that 3 epochs is enough training by ensuring that the loss curve flattens within 3 epochs of training.

5.3.2 Experiment 2: layer-by-layer reinitialization

For each finetuning task, we run 12 experiments. Each experiment is to reinitialize each layer and beyond. Our reinitialization uses $scipy.stats.truncnorm.rvs$ with -0.02 and 0.02 as the range. For each experiment, we run for 3 epochs after reinitializing the relevant layers.

5.3.3 Experiment 3: reinitializing attention heads

For each finetuning task, we run 48 (12 attention heads * 4 layers) experiments. For instance, one run would be reinitializing attention head 6 in layer 12 only for the CoLA task. We would use the BERT pretrained weights for the rest of the other attention heads and layers. Then we use this modified pretrained model to finetune on the GLUE task.

5.4 Results

5.5 Experiment 1: finetuning on multiple tasks

For this experiment, we run a series of experiments:

- CoLA as the final task
  - BERT pretrained layers → CoLA finetuning
  - BERT pretrained layers → MRPC finetuning → CoLA finetuning
  - BERT pretrained layers → RTE finetuning → CoLA finetuning
  - BERT pretrained layers → WNLI finetuning → CoLA finetuning
- MRPC as the final task
  - BERT pretrained layers → MRPC finetuning
  - BERT pretrained layers → CoLA finetuning → MRPC finetuning
  - BERT pretrained layers → RTE finetuning → MRPC finetuning
  - BERT pretrained layers → WNLI finetuning → MRPC finetuning
- RTE as the final task
  - BERT pretrained layers → RTE finetuning
  - BERT pretrained layers → CoLA finetuning → RTE finetuning
  - BERT pretrained layers → MRPC finetuning → RTE finetuning
  - BERT pretrained layers → WNLI finetuning → RTE finetuning
- WNLI as the final task
  - BERT pretrained layers → WNLI finetuning
  - BERT pretrained layers → CoLA finetuning → WNLI finetuning
  - BERT pretrained layers → MRPC finetuning → WNLI finetuning
  - BERT pretrained layers → RTE finetuning → WNLI finetuning

We obtain the following results:

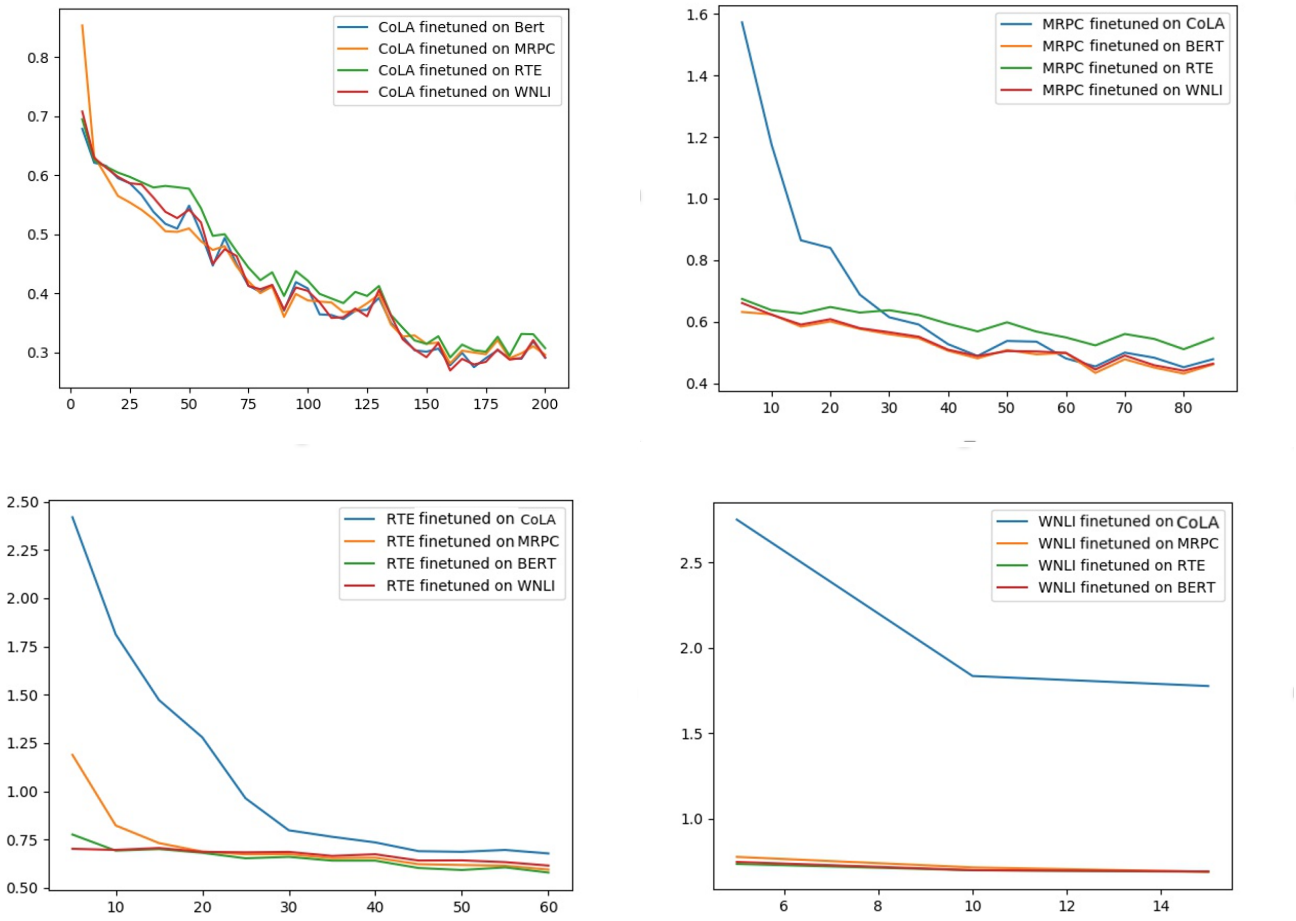

Figure 1: Loss values plotted against number of iterations during different finetuning tasks on different loaded weights

	CoLA	MRPC	RTE	WNLI	Metric
CoLA	0.526	0.523	**0.526**	**0.536**	mcc
MRPC	0.757	0.776	0.732	**0.786**	acc
RTE	0.462	0.610	0.635	0.588	acc
WNLI	0.408	0.394	0.464	0.549	acc

Figure 2: The column heading is the intermediate task and the row heading is the final task. The highest value for each final task is shaded. The values that are higher than the baseline (finetuned on only BERT pretrained layers) are bolded. The diagonal scores are the baseline scores (both the intermediate and final tasks are the same).

5.6 Experiment 2: layer-by-layer reinitialization

For this experiment, we ran layer-by-layer reinitialization on the tasks RTE, SQuAD, STS-B, CoLA and MRPC. The plots are shown below. The "layer 13" result represent not reinitializing any layer.

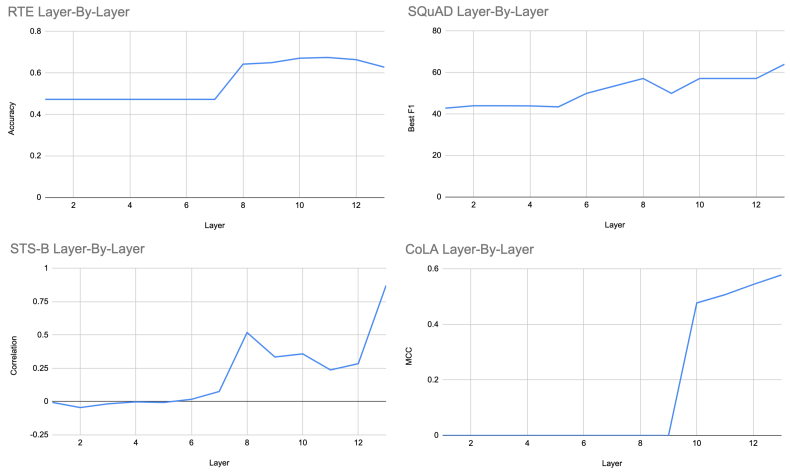

Figure 3: Reinitialized Layer-By-Layer Performance for Tasks. SQuAD plot generated with 10% dataset sampling.

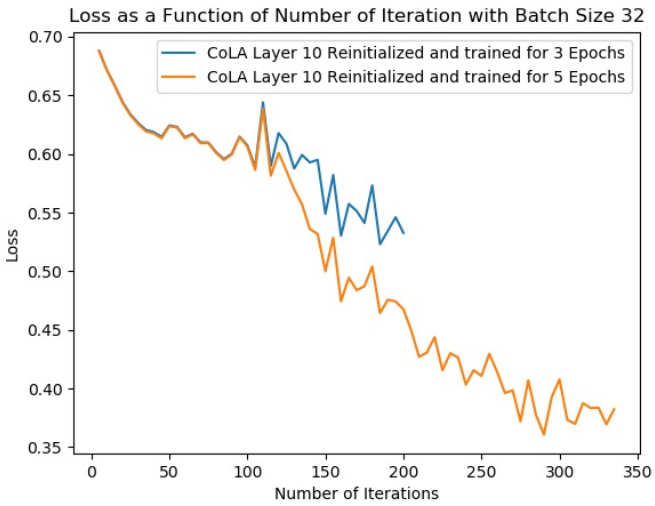

Figure 4: Loss as a function of number of iterations

For MRPC, we noticed that the performance does not change with reinitializing different layers.

	No Reinitialization	Reinitialize any k layers
Accuracy & F1	0.812	0.7480253018237863
F1	0.848	0.812
Accuracy	0.776	0.683

5.7 Experiment 3: reinitializing attention heads

For this experiment, we reinitialize each attention head per layer and separately finetune on two tasks. The baseline is not reinitializing any attention head in any layer. We use correlation to evaluate task STS-B and Matthew's correlation coefficient for task CoLA. We obtain our baseline score without any reinitialization and use the reinitialization experiment scores subtracted by the baseline scores to get the change in performance, as shown in below figures.

For CoLA, the base line correlation score is 0.526. Figure 5 shows the difference between the baseline performance and each attention head experiment:

	Head 1	**Head 2**	**Head 3**	**Head 4**	**Head 5**	**Head 6**	**Head 7**	**Head 8**	**Head 9**	**Head 10**	**Head 11**	**Head 12**
Layer 1	0.008	-0.023	-0.005	-0.015	0.01	-0.018	-0.015	-0.014	-0.026	-0.005	0.011	-0.007
Layer 2	-0.016	-0.007	-0.004	-0.045	-0.015	-0.031	-0.018	-0.043	-0.047	0	-0.016	-0.017
Layer 11	0.002	0.013	0.002	-0.013	0.02	0	0.002	0.007	0.002	0.013	0.013	0.015
Layer 12	0.008	-0.002	-0.002	-0.002	0.002	0.002	0	0.005	0.005	0.016	0.002	0.005

Figure 5: The difference between baseline and each attention head reinitialization experiment for CoLA fine tuning task. Most positive differences are highlighted in green and most negative differences are highlighted in red.

For STS-B, the baseline Matthew's Correlation Coefficient score is 0.872. Figure 6 shows the difference between the baseline performance and each attention head experiment for task STS-B:

	Head 1	**Head 2**	**Head 3**	**Head 4**	**Head 5**	**Head 6**	**Head 7**	**Head 8**	**Head 9**	**Head 10**	**Head 11**	**Head 12**
Layer 1	0.002	0.002	-0.002	-0.003	-0.003	-0.007	-0.004	-0.001	-0.007	-0.001	-0.005	-0.003
Layer 2	-0.001	-0.001	-0.001	-0.004	-0.002	-0.006	-0.003	-0.002	-0.023	-0.001	-0.004	-0.001
Layer 11	-0.003	0.003	0.004	-0.002	0	0.003	0	-0.003	-0.002	0.002	0	0.003
Layer 12	0	-0.002	0.003	-0.004	0.001	0.001	0	0.005	0	0	0.001	0.004

Figure 6: The difference between baseline and each attention head reinitialization experiment for STS-B fine tuning task. Most positive differences are highlighted in green and most negative differences are highlighted in red.

6 Analysis

6.1 Experiment 1: finetuning on multiple tasks

For CoLA as the final task, using WNLI as the intermediate task achieves a higher Matthew's correlation cofficient (mcc) than the mcc obtained by using RTE as the intermediate task. However, both achieve a higher mcc score than the baseline mcc score of 0.526 (directly finetuned on the pretrained BERT model).

For MRPC as the final task, using WNLI as an intermediate task achieves a higher accuracy than the the baseline accuracy score of 0.777 (directly finetuned on the pretrained BERT model).

For RTE as the final task, directly finetuning on the pretrained BERT model achieves the highest accuracy of 0.635.

For WNLI as the final task, directly finetuning on the pretrained BERT model achieves the highest accuracy of 0.549.

These results suggest that the weights in the pretrained portion of the model change significantly during finetuning. We want to understand how those weights change (what new information is encoded in those weights and what information is loss after finetuning). We see that finetuning on another finetuned model produces comparable results to the baseline of finetuning on BERT base, with a few instances in which finetuning on another finetuned model actually improves performance. This suggests that the relationships among these tasks are similar enough to where finetuning on one task allows the model to learn new semantic or syntactic information that can be transferred to another task. Furthermore, the motivation for the experiments also comes from a common insight in human pedagogy: learning different tasks at once is better than learning one task at a time, since the different tasks allow the learner to appreciate the nuances and differences among the tasks and how they relate, a process called interleaving by Birnbaum et al. [2013].

Concretely, we propose that the BERT model should incorporate the WNLI task as part of its pretraining. We see that using WNLI as an intermediate task and each of the other tasks as the final task outperforms the baseline, which is the performance of not using WNLI as an intermediate task (i.e. directly finetuning on the final task). The only final task for which using WNLI as an intermediate task has a worse performance is RTE. Therefore, we argue that finetuning on the WNLI task teaches BERT information that is fundamentally important and transferable to other tasks, so it makes sense to incorporate the information gleaned from the WNLI task into the pretrained layers.

6.2 Experiment 2: layer-by-layer reinitialization

For the layer-by-layer reinitialization, we expected the evaluation metric to monotonically increase with the number of layers reinitialized, since upstream layers encode more low-level linguistic information as compared to downstream layers, as stated by Tenney et al. [2018]. However, we see several deviations from our expectation for all tasks that we experimented on:

- For RTE, we see that the performance when initializing from layers $k \in [8, 12]$ actually surpasses the performance when no layers are reinitialized. We also observe that performance is flat before reinitializing layer 7. We suspect this trend happens due to underfitting the data. To improve, we can increase the number of epochs during finetuning. However, after layer 11, the performance starts to decrease, suggesting overfitting, which we can correct with regularization.
- For SQuAD, we notice that performance mostly increases monotonically, except when reinitializing from layer $k = 9$, where performance dips. Since the change is not very dramatic, we attribute this anomaly to random noise. To be sure, we can rerun this experiment multiple times and take the median for each layer reinitialization.
- For STS-B, we notice that correlation is near zero when we reinitialize all layers. The correlation stays near zero when we incrementally add back pretrained layers. When we add back the 8th pretrained layer and finetune, we see a spike in performance, with correlation equal to 0.5. This jump suggests that layer 8 contains important information that cannot be easily relearned by BERT during finetuning. Afterwards, the performance halves, and stays at this level until no layers are reinitialized, in which the performance triples.

- For CoLA, we observe that performance is flat until adding back layer 10, where it jumps to 0.6. After adding back layer 10, the performance only slowly increases when adding back more layers. Thus, not having the information from pretrained layer 10 prevents all other configurations from performing better than random.

The tasks all exhibit random guessing behavior when all the layers are reinitialized. No performance curve suggests that all layers contain equally important information for the finetuning task. For the RTE and SQuAD curves, we see that the upstream layers mostly have negligible effect on improving the performance over random guessing until the middle layers (layer 8 for RTE and layer 5 for SQuAD), where the performance increases and then levels off for the downstream layers. For STS-B, the performance spikes at level 8, but then decreases until the last level, where it spikes up. Somehow, the information encoded in layers 9 through 11 actually hinder the model from performing well. A further experiment could be to delete layers 9 through 11 to see if performance improves. Lastly, the CoLA curve suggests that layers 1 through 8 do not encode important information for the task, so a further experiment could be to remove those layers and finetune again.

6.3 Experiment 3: reinitializing attention heads

Figures 7 and 8 are the summaries for the difference between baseline and attention head reinitialization performance for CoLA and STS-B tasks at the layer-level. If we reinitialize any attention head in layers 1 or 2, the performance, up to the 75th percentile, will decrease. The reason may be that if we lose the information encoded in some pretrained attention heads in the upstream layers, the downstream layers may not work well either.

Another interesting finding is that reinitializing any attention head in the downstream layers improves the model performance. This improvement may be because reinitializing the heads adds noise to the model and reduces overfitting. We therefore propose that the attention heads in the downstream pretrained layers should be reinitialized.

	Count	Mean	Std Dev	Min	25%	50%	75%	Max
Layer 1	12	-0.008	0.013	-0.026	-0.016	-0.011	-0.001	0.011
Layer 2	12	-0.021	0.016	-0.047	-0.034	-0.016	-0.013	0.0002
Layer 11	12	0.006	0.009	-0.013	0.002	0.005	0.013	0.020
Layer 12	12	0.003	0.005	-0.002	-0.0007	0.002	0.005	0.016

Figure 7: CoLA attention heads: Difference by layer

	Count	Mean	Std Dev	Min	25%	50%	75%	Max
Layer 1	12	-0.003	0.003	-0.007	-0.004	-0.003	-0.001	0.002
Layer 2	12	-0.004	0.006	-0.023	-0.004	-0.002	-0.001	-0.001
Layer 11	12	0.0003	0.003	-0.003	-0.002	0.0001	0.003	0.004
Layer 12	12	0.0009	0.002	-0.004	-0.0004	0.001	0.002	0.005

Figure 8: STS-B attention heads: Difference by layer

Figures 9 and 10 are the summaries for the attention-head level difference for CoLA and STS-B. From the two figures, we can see that reinitializing attention heads 4 or 9, no matter for which layer or which task, will decrease the performance. This indicates that attention head 4 and 9 contains some important information for CoLA and STS-B tasks that cannot be easily relearned in a few epochs. Moreover, reinitializing attention head 12 improves the performance of the two finetuning tasks on average. Furthermore, different attention heads play different roles in different tasks. For example, reinitializing attention head 1 will increase the performance for CoLA but decrease the performance for STS-B on average, which means STS-B relies more on the information encoded in attention head 1 than CoLA does.

	Head 1	Head 2	Head 3	Head 4	Head 5	Head 6	Head 7	Head 8	Head 9	Head 10	Head 11	Head 12
Count	4	4	4	4	4	4	4	4	4	4	4	4
Mean	0.0009	-0.005	-0.002	-0.019	0.004	-0.011	-0.007	-0.011	-0.016	0.006	0.002	-0.001
Std Dev	0.011	0.015	0.003	0.018	0.015	0.016	0.01	0.023	0.025	0.01	0.013	0.014
Min	-0.016	-0.023	-0.005	-0.045	-0.015	-0.031	-0.018	-0.043	-0.047	-0.005	-0.016	-0.017
25%	-0.001	-0.011	-0.004	-0.023	-0.001	-0.021	-0.016	-0.021	-0.032	-0.001	-0.001	-0.01
50%	0.005	-0.005	-0.003	-0.014	0.006	-0.009	-0.007	-0.004	-0.012	0.006	0.007	-0.001
75%	0.008	0.001	-0.001	-0.01	0.013	0.0007	0.0005	0.006	0.003	0.013	0.012	0.007
Max	0.008	0.013	0.002	-0.002	0.02	0.002	0.002	0.007	0.005	0.016	0.013	0.015

Figure 9: CoLA: Difference summary at attention head level

	Head 1	Head 2	Head 3	Head 4	Head 5	Head 6	Head 7	Head 8	Head 9	Head 10	Head 11	Head 12
Count	4	4	4	4	4	4	4	4	4	4	4	4
Mean	-0.0008	0.0004	0.001	-0.003	-0.001	-0.002	-0.001	-0.0006	-0.008	0.00002	-0.002	0.0005
Std Dev	0.002	0.002	0.003	0	0.002	0.005	0.002	0.004	0.01	0.002	0.003	0.003
Min	-0.003	-0.002	-0.002	-0.004	-0.003	-0.007	-0.004	-0.003	-0.023	-0.001	-0.005	-0.003
25%	-0.001	-0.001	-0.001	-0.004	-0.003	-0.006	-0.003	-0.002	-0.011	-0.001	-0.004	-0.002
50%	-0.001	0.0002	0.0008	-0.003	-0.001	-0.002	-0.001	-0.002	-0.004	-0.0002	-0.002	0.001
75%	-0.00009	0.002	0.003	-0.003	-0.00006	0.002	0.0007	-0.00004	-0.001	0.001	0.0001	0.003
Max	0.002	0.003	0.004	-0.002	0.001	0.003	0.0009	0.005	-0.0003	0.002	0.001	0.004

Figure 10: STS-B: Difference summary at attention head level

7 Conclusion and Future Work

We have several proposals for modifying the pretrained layers of BERT to yield better improvements. First, we propose that the pretrained layers be trained on the WNLI task, as it yields better performance for any other task that is finetuned on the model afterwards. We also saw preliminary results in which interleaving different tasks on the model improved the results. Secondly, we see that reinitializing some attention heads in the downstream layers improve model performance, so we should reinitialize them in the pretrained model. However, since attention head 4 and 9 play an important role in finetuning tasks, we should not reinitialize these two attention heads.

7.1 Future Work

In our layer-by-layer reinitialization of our BERT pretrained model, we have behaviors that suggest underfitting. Specifically, we see in Figure 10 the loss of the BERT pretrained model with layer 10 reinitialized continues to decrease beyond the 3 epochs, the number of epochs that was used to obtain the baseline for the various finetuning tasks. Thus, with access to much more compute power and time, we hope to to run the reinitialized model until the loss during training plateaus and meets some predefined convergence criteria, rather than limiting the number of epochs for which the model trains.

We also want to run the experiments multiple times or do bootstrap resampling to determine the statistical significance of our results.

Furthermore, with our experiments in which we finetuned on multiple tasks, we saw great variations in the results. This variation was expected as the existing literature containing these standardized datasets reported significant variations due to the small size of these data sets. Thus, with more time and resources, we hope to run each experiment 10 times and take the median, rather than running the experiment twice and taking the better score. Further, we are interested in finetuning on models that have been through several different finetuning tasks, instead of just having one intermediate finetuning task.

8 Authorship Statement

All authors contributed equally to this project.

Estimated time spent on this project: 457 hours.

For our code, we built off of the Transformers library by HuggingFace.

References

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Association for Computational Linguistics (ACL)*, 2018.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? volume abs/1905.10650, 2019. URL `http://arxiv.org/abs/1905.10650`.

Amir R. Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. volume abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Monica S Birnbaum, Nate Kornell, Elizabeth Ligon Bjork, and Robert A Bjork. Why interleaving enhances inductive learning: The roles of discrimination and retrieval. *Memory & cognition*, 41(3):392–402, 2013.