Investigating Transferability in Pretrained Language Models

Alex Tamkin[†] Stanford University
Trisha Singh Stanford University
Davide Giovanardi Stanford University
Noah Goodman Stanford University

Abstract

How does language model pretraining help transfer learning? We consider a simple ablation technique for determining the impact of each pretrained layer on transfer task performance. This method, *partial reinitialization*, involves replacing different layers of a pretrained model with random weights, then finetuning the entire model on the transfer task and observing the change in performance. This technique reveals that in BERT, layers with high probing performance on downstream GLUE tasks are *neither necessary nor sufficient* for high accuracy on those tasks. Furthermore, the benefit of using pretrained parameters for a layer varies dramatically with finetuning dataset size: parameters that provide tremendous performance improvement when data is plentiful may provide negligible benefits in data-scarce settings. These results reveal the complexity of the transfer learning process, highlighting the limitations of methods that operate on frozen models or single data samples.

1 Introduction

Despite the striking success of transfer learning in NLP, remarkably little is understood about how these pretrained models improve downstream task performance. Recent work on understanding deep NLP models has centered on *probing*, a methodology that involves training classifiers for different tasks on model representations (Alain and Bengio, 2016; Conneau et al., 2018; Hupkes et al., 2018; Liu et al., 2019; Tenney et al., 2019a,b; Goldberg, 2019; Hewitt and Manning, 2019). While probing aims to uncover what a network has already learned, a major goal of machine learning is *transfer*: systems that build upon what they have learned to expand what they *can* learn. Given that most recent models are updated end-to-end during finetuning (e.g. Devlin et al., 2019; Howard and Ruder, 2018; Radford et al., 2019), it is unclear how, or even whether, the knowledge uncovered by probing contributes to these models' transfer learning success.

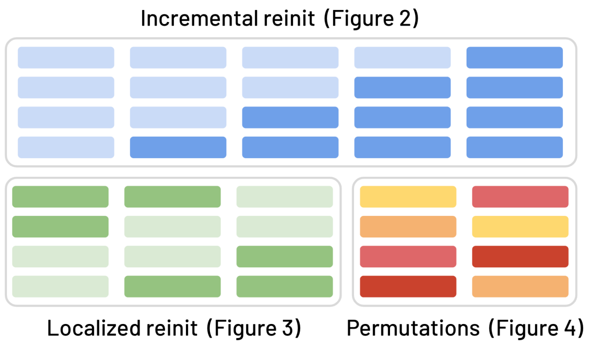

Figure 1: **The three experiments we explore.** Lighter shades indicate randomly reinitialized layers, while darker shades indicate layers with BERT parameters. For layer permutations, all layers hold BERT parameters, what changes between trials is their order. In all three experiments, the entire model is finetuned end-to-end on the GLUE task.

In a sense, probing can be seen as quantifying the *transferability of representations* from one task to another, as it measures how well a simple model (e.g., a softmax classifier) can perform the second task using only features from a model trained on the first. However, when pretrained models are finetuned end-to-end on a downstream task, what is transferred is not the features from each layer of the pretrained model, but its *parameters*, which define a sequence of functions for processing representations. Critically, these functions and their interactions may shift considerably during training, potentially enabling higher performance despite not initially extracting features correlated with this task. We refer to this phenomenon of how layer parameters from one task can help transfer learning

[†] atamkin@stanford.edu

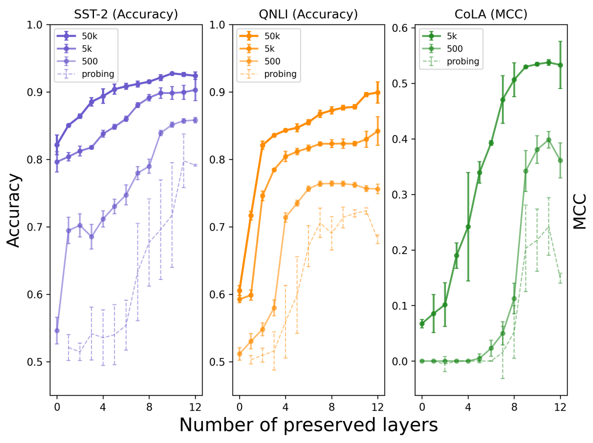

Figure 2: **The benefit of using BERT parameters instead of random parameters at a particular layer varies dramatically depending on the size of the finetuning dataset. However, as finetuning dataset size decreases, the curves align more closely with probing performance at each layer.** Solid lines show finetuning results after reinitializing all layers past layer k in BERT-Base. 12 shows the full BERT model, while 0 shows a model with all layers reinitialized. Line darkness indicates subsampled dataset size. The dashed lines show probing performance at each layer. Error bars are 95% CIs.

on another task as *transferability of parameters*.

In this work, we investigate a methodology for measuring the transferability of different layer parameters in a pretrained language model to different transfer tasks, using BERT (Devlin et al., 2019) as our subject of analysis. Our methods, described more fully in Section 2 and Figure 1, involve *partially reinitializing* BERT: replacing different layers with random weights and then observing the change in task performance after finetuning the entire model end-to-end. Compared to possible alternatives like freezing parts of the network or removing layers, partial reinitialization enables fairer comparisons by keeping the network's architecture and capacity constant between trials, changing only the parameters at initialization. Through experiments across different layers, tasks, and dataset sizes, this approach enables us to shed light on multiple dimensions of the transfer learning process: Are the early layers of the network more important than later ones for transfer learning? Do individual layers become more or less critical depending on the task or amount of finetuning data? Does the position of a particular layer within the network matter, or do its parameters aid optimization regardless of where they are in the network?

We find that when finetuning on a new task:

1. Transferability of BERT layers varies dramatically depending on the amount of finetuning data available. Thus, claims that certain layers are universally responsible or important for learning certain linguistic tasks should be treated with caution. (Figure 2)

2. Transferability of BERT layers is not in general predicted by the layer's probing performance for that task. However, as finetuning dataset size decreases, the two quantities exhibit a greater correspondence. (Figure 2, dashed lines)

3. Even holding dataset size constant, the most transferable BERT layers differ by task: for some tasks, only the early layers are important, while for others the benefits are more distributed across layers. (Figure 3)

4. Reordering the pretrained BERT layers before finetuning decreases downstream accuracy significantly, confirming that pretraining does not simply provide better-initialized individual layers; instead, transferability through learned interactions *across layers* is crucial to the success of finetuning. (Figure 4)

2 How many pretrained layers are necessary for finetuning?

Our first set of experiments aims to uncover how many pretrained layers are sufficient for accurate learning of a downstream task. To do this, we perform a series of **incremental reinitialization** experiments, where we reinitialize all layers after the kth layer of BERT-Base, for values $k \in \{0, 1, \dots 12\}$, replacing them with random weights. We then finetune the entire model end-to-end on the target task. Note that $k = 0$ corresponds to a BERT model with all layers reinitialized, while $k = 12$ is the original BERT model. We do not reinitialize the BERT word embeddings. As BERT uses residual connections (He et al., 2016) around layers, the model can simply learn to ignore any of the reinitialized layers if they are not helpful during finetuning.

We use the BERT-Base uncased model, implemented in PyTorch (Paszke et al., 2019) via the Transformers library (Wolf et al., 2019). We finetune the network using Adam (Kingma and Ba, 2015), with a batch size of 8, a learning rate of 2e-5, and default parameters otherwise. More details about reinitialization, training, statistical significance, and other methodological choices can be found in the Appendix. We conduct our experiments on three English language tasks from the GLUE benchmark, spanning the domains of sentiment, reasoning, and syntax (Wang et al., 2018):

SST-2 Stanford Sentiment Treebank involves binary classification of a single sentence from a movie review as positive or negative (Socher et al., 2013).

QNLI Question Natural Language Inference is a binary classification task derived from SQuAD (Rajpurkar et al., 2016; Wang et al., 2018). The task requires determining whether for a given (QUESTION, ANSWER) pair the QUESTION is answered by the ANSWER.

CoLA The Corpus of Linguistic Acceptability is a binary classification task that requires determining whether a single sentence is linguistically acceptable (Warstadt et al., 2019).

Because pretraining appears to be especially helpful in the small-data regime (Peters et al., 2018), it is crucial to isolate task-specific effects from data quantity effects by controlling for finetuning dataset size. To do this, we perform our incremental reinitializations on randomly-sampled subsets of the data: 500, 5k, and 50k examples (excluding 50k for CoLA, which contains only 8.5k examples). The 5k subset size is then used as the default for our other experiments. To ensure that an unrepresentative sample is not chosen by chance, we run multiple trials with different subsamples. Confidence intervals produced through multiple trials also demonstrate that trends hold regardless of intrinsic task variability.

While similar reinitialization schemes have been explored by Yosinski et al. (2014); Raghu et al. (2019) in computer vision and briefly by Radford et al. (2019) in an NLP context, none investigate these data quantity- and task-specific effects.

Figure 2 shows the results of our incremental reinitialization experiments. These results show that the transferability of a BERT layer varies dramatically based on the finetuning dataset size. Across all but the 500 example trials of SST-2, a more specific trend holds: earlier layers provide more of an improvement on finetuning performance when the finetuning dataset is large. This trend suggests that larger finetuning datasets may enable the network to learn a substitute for the parameters in the middle and later layers. In contrast, smaller datasets may leave the network reliant on existing feature processing in those layers. However, across all tasks and dataset sizes, it is clear that the pretrained parameters by themselves do not determine the impact they will have on finetuning performance: instead, a more complex interaction occurs between the parameters, optimizer, and the available data.

3 Does probing predict layer transferability?

What is the relationship between transferability of representations, measured by probing, and transferability of parameters, measured by partial reinitialization? To compare, we conduct probing experiments for our finetuning tasks on each layer of the pretrained BERT model. Our probing model averages each layer's hidden states, then passes the pooled representation through a linear layer and softmax to produce probabilities for each class. These task-specific components are identical to those in our reinitialization experiments; however, we keep the BERT model's parameters frozen when training our probes.

Our results, presented in Figure 2 (dashed lines), show a significant difference between the layers with the highest probing performance and reinitialization curves for the data-rich settings (darkest solid lines). For example, the probing accuracy on all tasks is near chance for the first six layers. Despite this, these early layer parameters exhibit significant transferability to the finetuning tasks: preserving them while reinitializing all other layers enables large gains in finetuning accuracy across tasks. Interestingly, however, we observe that the smallest-data regime's curves are much more similar to the probing curves across all tasks than the larger-data regimes. Smaller finetuning datasets enable fewer updates to the network before overfitting occurs; thus, it may be that finetuning interpolates between the extremes of probing (no data) and fully-supervised learning (enough data to completely overwrite the pretrained parameters). We leave a more in-depth exploration of this connection to future work.

4 Which layers are most useful for finetuning?

While the incremental reinitializations measure each BERT layer's incremental effect on transfer

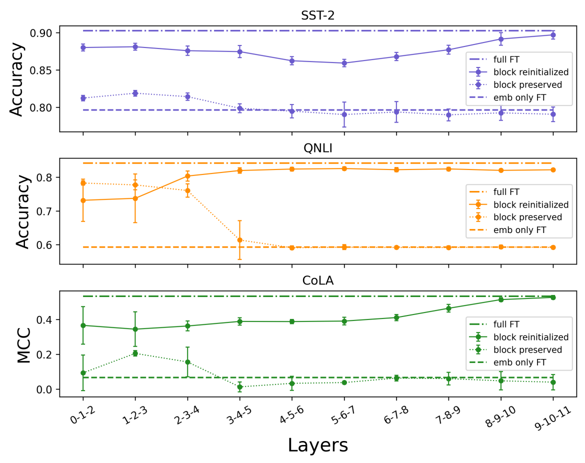

Figure 3: **Early layers provide the most QNLI gains, but middle ones yield an added boost for CoLA and SST-2.** Finetuning results for 1) reinitializing a consecutive three-layer block ("block reinitialized") and 2) reinitializing all other layers ("block preserved"). Dashed horizontal lines show the finetuning performance of the full BERT model and the performance of a model with only embedding parameters preserved. Finetuning trials with 5k examples. Error bars are 95% CIs.

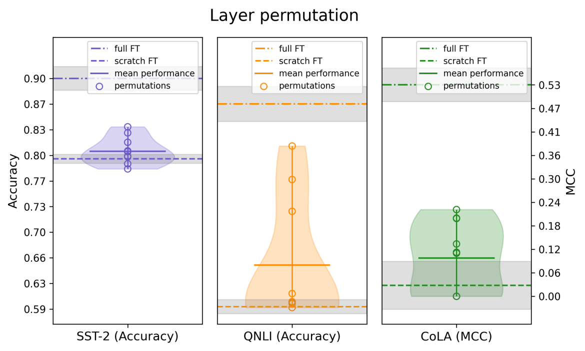

Figure 4: **Changing the order of pretrained layers harms finetuning performance significantly.** Dashed lines mark the performance of the original BERT model and the randomly-initialized model (surrounded by $\pm 2\sigma$ error bars). Circles denote finetuning performance for different layer permutations, while the solid line denotes the mean across runs (with 95% CIs). The curved shaded region is a kernel density plot, which illustrates the distribution of outcomes. Finetuning trials with 5k examples.

learning, they do not assess each layer's contribution in isolation, relative to either the full BERT model or an entirely reinitialized model. Measuring this requires eliminating the *number* of pretrained layers as a possible confounder. To do so, we conduct a series of **localized reinitialization** experiments, where we take all blocks of three consecutive layers and either 1) *reinitialize* those layers or 2) *preserve* those layers while reinitializing the others in the network.[1] These localized reinitializations help determine the extent to which BERT's different layers are either necessary (performance decreases when they are removed) or sufficient (performance is higher than random initialization when they are kept) for a specific level of performance. Again, BERT's residual connections permit the model to ignore reinitialized layers' outputs if they harm finetuning performance.

These results, shown in Figure 3, demonstrate that the earlier layers appear to be generally more helpful for finetuning relative to the later layers, even when controlling for the amount of finetuning data. However, there are strong task-specific effects: SST-2 appears to be particularly damaged by removing middle layers, while the effects on CoLA are distributed more uniformly. The effects on QNLI appear to be concentrated almost entirely in the first four layers of BERT—suggesting opportunities for future work on whether sparsity of this sort indicates the presence of easy-to-extract features correlated with the task label. These results support the hypothesis that different kinds of feature processing learned during BERT pretraining are helpful for different finetuning tasks, and provide a new way to gauge similarity between different tasks.

5 How vital is the ordering of pretrained layers?

We also investigate whether the success of BERT depends mostly on learned *inter-layer phenomena*, such as learned feature processing pipelines (Tenney et al., 2019a), or *intra-layer phenomena*, such as a learned feature-agnostic initialization scheme which aid optimization (e.g. Glorot and Bengio, 2010). To approach this question, we perform several **layer permutation** experiments, where we randomly shuffle the order of BERT's layers before finetuning. The degree that finetuning performance is degraded in these runs indicates the extent to which BERT's finetuning success is dependent on a learned composition of feature processors, as opposed to providing better-initialized individual layers which would help optimization anywhere in the network.

These results, plotted in Figure 4, show that scrambling BERT's layers reduces their finetuning

[1]See the Appendix for more discussion and experiments where only one layer is reinitialized.

ability to not much above a randomly-initialized network, on average. This decrease suggests that BERT's transfer abilities are highly dependent on the intra-layer interactions learned during pretraining.

We also test for correlation of performance between tasks. We do this by comparing task-pairs for each permutation, as we use the same permutation for the nth run of each task. The high correlation coefficients for most pairs shown in Table 1 suggest that BERT finetuning relies on similar inter-layer structures across tasks.

Tasks compared	Spearman	Pearson
SST-2, QNLI	0.72 (0.02)	0.46 (0.18)
SST-2, CoLA	0.74 (0.02)	0.77 (0.01)
QNLI, CoLA	0.83 (0.00)	0.68 (0.03)

Table 1: **Specific permutations of layers have similar impacts on finetuning across tasks.** Paired correlation coefficients between task performances for the same permutations. Two-sided p-value in parentheses (N=10).

6 Conclusion

We present a set of experiments to better understand how the different pretrained layers in BERT influence its transfer learning ability. Our results reveal the unique importance of transferability of parameters to successful transfer learning, distinct from the transferability of fixed representations assessed by probing. We also disentangle important factors affecting the role of layers in transfer learning: task vs. quantity of finetuning data, number vs. location of pretrained layers, and presence vs. order of layers.

While probing continues to advance our understanding of linguistic structures in pretrained models, these results indicate that new techniques are needed to connect these findings to their potential impacts on finetuning. The insights and methods presented here are one contribution toward this goal, and we hope they enable more work on understanding why and how these models work.

Acknowledgements

We would like to thank Dan Jurafsky, Pranav Rajpurkar, Shyamal Buch, Isabel Papadimitriou, John Hewitt, Peng Qi, Kawin Ethayarajh, Nelson Liu, and Jesse Michel for useful discussions and comments on drafts. This work was supported in part by DARPA under agreement FA8650-19-C-7923.

References

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

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

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

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Kaiming He, Ross Girshick, and Piotr Dollár. 2019. Rethinking imagenet pre-training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4918–4927.

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

John Hewitt and Christopher D Manning. 2019. 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.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

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.

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

Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. 2019. Transfusion: Understanding transfer learning for medical imaging. In *Advances in neural information processing systems*, pages 3347–3357.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

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*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam 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*.

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

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

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

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

A Code

Our code is available at `https://github.com/dgiova/bert-lm-transferability`.

B Reinitialization

We reinitialize all parameters in each layer, except those for layer normalization (Ba et al., 2016), by sampling from a truncated normal distribution with $\mu = 0, \sigma = 0.02$ and truncation range $(-0.04, 0.04)$. For the layer norm parameters, we set $\beta = 0, \gamma = 1$. This matches how BERT was initialized (see the original BERT code on GitHub and the corresponding TensorFlow documentation).

C Subsampling, number of trials, and error bars

The particular datapoints subsampled can have a large impact on downstream performance, especially when data is scarce. To capture the full range of outcomes due to subsampling, we randomly sample a different dataset for each trial index. Due to this larger variation when data is scarce, we perform 50 trials for the experiments with 500 examples, while we perform three trials for the other incremental reinitialization experiments. A scatterplot of the 500-example trials is shown in Figure 5. For the localized reinitialization experiments, we perform ten trials each.

Error bars shown on all graphs in the main text are 95% confidence intervals calculated with a t-distribution.

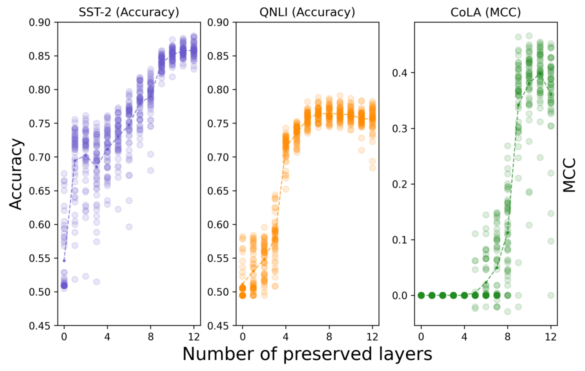

Figure 5: Finetuning results after reinitializing all layers past layer k in BERT-Base. 12 shows the full BERT model, while 0 shows a model with all layers reinitialized. Scatterplot of 50 trials per layer shown for subsampled dataset size 500. Dotted line shows the mean.

D Localized reinitializations of single layers

We also experiment with performing our localized reinitialization experiments at the level of a single layer. To do so, we perform three trials of reinitializing each layer $k \in \{1 \ldots 12\}$ and then finetuning on each of the three GLUE tasks. Our results are plotted in Figure 6. Interestingly, we observe little effect on finetuning performance from reinitializing each layer (except for reinitializing the first layer on CoLA performance). This lack of effect suggests either redundant information between layers or that the "interface" exposed by the two neighboring layers somehow beneficially constrains optimization.

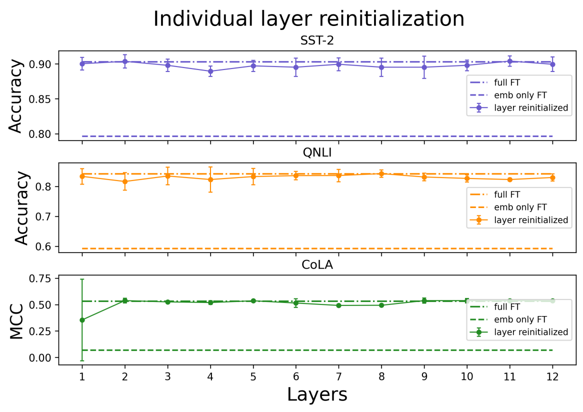

Figure 6: Performance on finetuning tasks after reinitializing an individual layer of BERT. Error bars are ± 2 standard deviations.

E Number of finetuning epochs

He et al. (2019) found that much or all of the performance gap between an ImageNet-pretrained model and a model trained from random initialization could be closed when the latter model was trained for longer. To evaluate this, we track validation losses up to ten epochs in our incremental experiments, for $k \in \{0, 6, 12\}$ across all tasks and for 500 and 5k examples. We find minimal effects of training longer than three epochs for the subsamples of 5k, but find improvements of several percentage points for training for five epochs for the trials with 500 examples. Thus, for the trials of 500 in Figure 2, we train for five epochs, while training for three epochs for all other trials. We train our probing experiments (8 trials per layer) with early stopping for a maximum of 40 epochs on the full dataset.

F Higher learning rate for reinitialized layers

In their reinitialization experiments on a convolutional neural network for medical images, Raghu et al. (2019) found that a 5x larger rate on the reinitialized layers enabled their model to achieve higher finetuning accuracy. To evaluate this possibility in our setting, we increase the learning rate by a factor of five for the reinitialized layers. The results for our incremental reinitializations are plotted in Figure 7. A higher learning rate appears to increase the variance of the evaluation metrics while not improving performance. Thus, we keep the learning rate the same across layers.

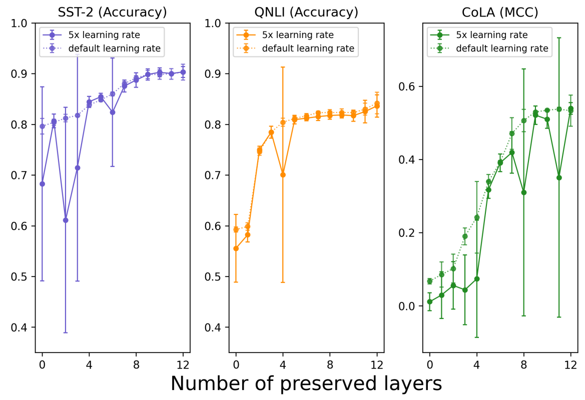

Figure 7: Finetuning the reinitialized layers with a larger learning rate does not improve finetuning performance. Error bars are ± 2 standard deviations.

G Layer norm

Because the residual connections around each sublayer in BERT are of the form $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, reinitializing a particular layer neutralizes the effect of the last layer norm application from the previous layer in a way that cannot be circumvented through the residual connections. However, for brevity we simply refer to “reinitializing a layer” in this paper.

We also assessed whether preserving the layer norm parameters in each layer might aid optimization. To do so, we preserved these parameters in our incremental trials with 5k examples. These trials are plotted in Figure 8, and demonstrate that preserving layer norm does not aid (and may even harm) finetuning of reinitialized layers.

H Dataset descriptions and statistics

We display more information about the finetuning datasets, including the full size of the datasets, in

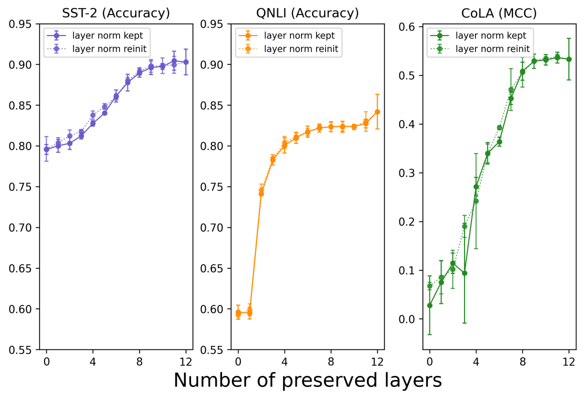

Figure 8: Preserving the layer norm parameters when reinitializing each layer does not improve finetuning performance. Error bars are ± 2 standard deviations.

Table 2.

I Additional experimental information

I.1 Link to data

Scripts to download the GLUE data can be found at `https://github.com/nyu-mll/jiant/blob/master/scripts/download_glue_data.py`.

I.2 Computing infrastructure

All experiments were run on single Titan XP GPUs.

I.3 Model

We use the BERT-Base uncased model (110 million parameters) from `https://huggingface.co/transformers/pretrained_models.html`.

I.4 Average runtime

Average runtime for each approach:

1. **500 incremental**: 0.3 min / epoch * 5 epochs / trial * 50 trials / layer * 12 layers / task * 3 tasks $\approx$ 45 GPU-hrs
2. **5k incremental**: 3 min / epoch * 3 epochs / trial * 3 trials / layer * 12 layers / task * 3 tasks $\approx$ 16 GPU-hrs.
3. **50k incremental**: 30 min / epoch * 3 epochs / trial * 3 trials / layer * 12 layers / task * 3 tasks $\approx$ 7 GPU-days.
4. **5k localized (block size 3)**: 3 min / epoch * 3 epochs / trial * 3 trials / layer * 10 layers / task * 3 tasks $\approx$ 14 GPU-hrs
5. **Probing**: 2.8 min / epoch * 40 epochs / trial * 8 trials / layer * 12 layers / task * 3 tasks

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

Task	# Train	# Val	Input, labels	Eval metric
SST-2	67k	872k	sentence, {positive, negative}	Accuracy
QNLI	105k	5.4k	(question, paragraph), {answer, non-answer}	Accuracy
CoLA	8.5k	1k	sentence, {acceptable, not acceptable}	MCC

$\approx$ 22 GPU-days. *Note: 2.8 min / epoch is an average across layers and tasks. Earlier layers take less time than later ones because layers after the target layer do not need to be computed.*

I.5 Evaluation method

To evaluate the performance of our method, we compute *accuracy* for SST-2 and QNLI and *Matthews Correlation Coefficient* (Matthews, 1975) for CoLA. We compute these metrics always on the official validation sets, which are 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, MCC is used, which is better suited for unbalanced binary classifiers (Warstadt et al., 2019). 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. $MCC = \frac{(TP \cdot TN)-(FP \cdot FN))}{\sqrt{(TP+FP)(TP+FN)(FP+TN)(TN+FN)}}$

I.6 Hyperparameters

We performed one experiment with a 5x learning rate and implemented early stopping to choose the number of epochs for the probing experiments.

For batch size and learning rate, we kept the default parameters for all tasks:

- Learning rate: 2e-5
- Batch size: 8