Learning by Prediction and Diversity with BERT

Stanford CS224N Default Project
Mentor:Rohan Taori, External Collaborators:None, Sharing project:N/A

Alex Lin
Stanford University
alexlin0@stanford.edu

Abstract

Pre-trained BERT (Devlin et al., 2018) has demonstrated incredible performances in numerous downstream language tasks when the model is fine-tuned using pre-trained weights on downstream tasks. Techniques to further improve BERT embeddings are desired to not only improve the performance of downstream tasks, but also to gain insight into how the BERT model behaves and explore avenues for further improvement of language models. This paper explores a number of techniques applied to BERT on four downstream tasks: sentiment classification, paraphrase detection, semantic textual similarity, and linguistic acceptability. The techniques explored include: per-layer learning rate, LoRA, further pre-training, and multitask fine-tuning (Sun et al., 2019) (Hu et al., 2021). We find that on difficult tasks with low performance such as sentiment classification, further pre-training has a big impact on performance, and we find that pre-training on different domain datasets and multitask fine-tuning can exhibit transfer learning properties by improving performance of related tasks. Additionally, we find that LoRA applied on a general pre-trained model achieves similar performance as full fine-tuning, but achieves much lower performance on a less general model that is pre-trained on task-specific data. The experiments from this paper support the prospects and potential of improved performance from scaling exhibited by many recent LLM research which pursue scaling model sizes, dataset diversity, and modalities diversity (Aghajanyan et al., 2023).

1 Introduction

One of the most distinguishing abilities of humans is the ability to communicate complex thoughts and tasks through natural language. There are a wide variety of complex tasks which can be expressed through natural language, and the field of natural language processing aims to build systems which can solve these tasks as well as, or even better, than humans.

Recent advances in the field have led to systems exhibiting incredible human-like behavior, such as ChatGPT (OpenAI). Most notably, the technique of pre-training through prediction (Devlin et al., 2018), along with the transformer architecture (Vaswani et al., 2017), have led to the advent of large language models which could be scaled up in model and dataset size seemingly indefinitely, leading to incredible emergent behavior and state-of-the-art performance in a variety of natural language processing tasks.

One of the earliest systems to demonstrate incredible performance at scale in numerous tasks using pre-training on transformers was BERT (Devlin et al., 2018). Utilizing the BERT model architecture and its pre-trained weights, this paper aims to gain insight into the impact of pre-training on performance and the ability of BERT to generalize well to multiple tasks by applying further pre-training and multitask fine-tuning to downstream tasks. Additionally, the LoRA technique (Hu et al., 2021) was explored to gain intuition into the generalizability of BERT's pre-trained weights to different tasks. Lastly, per-layer finetuning was applied to gain insight into the transformer architecture.

2 Related Work

Before the advent of transformers, ideas of applying pre-training to RNNs have been explored (Pasa, 2014). However, it is difficult to scale RNNs up due to difficulty in parallelizing computations and difficulty in maintaining long contexts as a result of the RNN model architecture. The advent of transformers and the attention mechanism (Vaswani et al., 2017) has been demonstrated to not only be able to acheive higher quality, but is also more parallelizable, thus overcoming the shortcomings of RNNs and becoming a mainstay in the field of machine learning.

Radford et al. (2018) applies the idea of pre-training on the transformer architecture. Pre-training is applied to a transformer decoder using the standard language modeling objective. This method achieves state-of-the-art results in many tasks when compared to previous state-of-the-art LSTM models.

BERT (Devlin et al., 2018) expands upon previous work in transformers by introducing a bi-directional pre-training technique by jointly conditioning on left and right contexts, called Masked LM. Using Masked LM, BERT trains a transformer encoder which allows the model to access both the left and right contexts when making predictions. This technique was applied to the largest model at the time, with the BERT large variant having 340M parameters.

Further research demonstrate that pre-training techniques and transformers continue to scale well with model size and dataset size. GPT-3 is a model with 175 billion parameters which not only demonstrates state-of-the-art results, but also demonstrates the emergent behavior of few-shot learning, which achieves performance, without fine-tuning nor any gradient descent update steps, that is close to the state-of-the-art in some tasks (Brown et al., 2020). Many similar models demonstrate empirical data which supports the continued scaling of models. However, more recently, there is research which aims to understand the empirical data for model scale in the form of developing scaling laws for language models based on controlled experiments while introducing data and task diversity in the form of multiple modalities (Aghajanyan et al., 2023).

3 Approach

We apply four techniques to BERT: Per-Layer Learning Rate (Pasa, 2014), Low Rank Adaptation (Hu et al., 2021), Further Pre-Training (Devlin et al., 2018), and Multitask Fine-Tuning (Pasa, 2014). We apply the four techniques to a baseline BERT model, with a final layer head for each downstream task.

3.1 Adapting BERT to Downstream Tasks

The downstream tasks chosen include: sentiment analysis, paraphrase detection, semantic textual similarity, and linguistic acceptability. Sentiment analysis is the task of classifying positive or negative sentiment of text into 5 discrete categories, from 1 to 5 in terms of positivity. Paraphrase detection is classifying whether two sentences are paraphrases of each other. Semantic textual similarity is determining the similarity, on a scale of 1 to 5, the similarity between two sentences. Linguistic acceptability is classifying whether a sentence is grammatically and semantically acceptable and reasonable. Datasets for these tasks are described in section 4.1

The approach to adapt the BERT model architecture to downstream tasks is to add a final layer head for each downstream task which will use the pooler output's classification hidden state (cls) as input. The final layer is a multilayer perceptron feedforward layer, with a single hidden layer of size $h = 256$ chosen arbitrarily and shared between all four downstream tasks, the GeLU activation function (Hendrycks and Gimpel, 2016), and a final linear layer for each downstream task.

For sentiment analysis, we use a final linear layer with 5 outputs, one for each sentiment. We apply softmax and use cross entropy loss between the one-hot label and the predicted categorical distribution.

For paraphrase detection, we use a final linear layer with 1 output. We use the binary cross entropy loss function between the true paraphrase label and the predicted output.

For semantic textual similarity, we use a final linear layer with 1 output with a mean-squared error loss function to predict a float value between 1 and 5.

For the paraphrase detection and semantic textual similarity tasks, input data consists of two sentences which are concatenated together with a [SEP] token before being fed into the model. This is to take advantage of self-attention across both sentences.

For linguistic acceptability, we use a final linear layer with 1 output with binary cross entropy loss between the true label and predicted output.

3.2 Baseline

The baseline will be three-fold: (1) the frozen pre-trained BERT model from the default project with a fine-tuned final layer for text classification, (2) a fully fine-tuned BERT model and final layer on each of the four tasks without any techniques applied, and (3) the same technique as 2, but with per-layer learning rate with a decay of 0.95 applied to BERT layers. All code for techniques and baselines mentioned are originally coded unless explicitly mentioned otherwise.

3.3 Per-Layer Learning Rate

To apply per-layer learning rate (Sun et al., 2019), we decay the learning rate of the BERT model across layers. The BERT model consists of $N = 14$ layers: the embedding layer, 12 attention layers, and a pooler layer. Given a learning rate lr, the learning rate for a given layer i, denoted lr_i, is reduced by a factor of β. $lr_{i-1} = \beta lr_i$. We chose $\beta = 0.95$, as used in Sun et al. (2019). The final layer heads will use the full learning rate lr. To observe the impact of each layer on performance, we run additional experiments which freeze all layers except for one and fine-tune the model to observe performance improvement over the frozen BERT baseline given one unfrozen layer.

3.4 Low Rank Adaptation (LoRA)

To apply LoRA (Hu et al., 2021), we froze all of the parameters of the original BERT model. Given a LoRA rank r, for each embedding matrix and weight matrix $W_i \in \mathbb{R}^{d \times k}$, we construct new parameters $A_i \in \mathbb{R}^{d \times r}$ and $B_i \in \mathbb{R}^{r \times k}$. The forward computation of the model will then take $W_i \leftarrow W_i + A_i B_i$, where A_i and B_i are trainable. We experimented with $r = 1, 2, 4, 8$ to evaluate the impact of LoRA rank on performance. The following LoRA implementation was utilized: https://github.com/ccctu/minLoRA.

3.5 Further Pre-Training

To apply further pre-training, we applied the MaskLM training technique (Devlin et al., 2018) on the input tokens from the datasets. For sentiment analysis and linguistic acceptability which only consists of one sentence, the MaskLM technique masks tokens in that sentence. For paraphrase detection and semantic textual analysis which consists of two sentences joined together by a [SEP] token, MaskLM is applied on both sentences, but never the [SEP] token, nor other special tokens. Further pre-training is run for each task with pre-training on its own task dataset and with pre-training on all four tasks datasets combined. For all tasks, both 15 and 30 epochs of pre-training are tested. For sentiment analysis, all epochs 1 through 30 are tested to evaluate the impact of pre-training on performance across epochs. After pre-training, we use the BERT model parameters as initialization parameters and fine-tune the model on a single downstream task and evaluate performance.

3.6 Multitask Fine-Tuning

For multitask fine-tuning, at each iteration, we use the following method: randomly uniformly sample a task i amongst all four tasks and perform a single mini-batch step for task i. For each epoch, 1000 mini-batch iterations were run, and 3 total epochs of multitask fine-tuning were run. After multitask fine-tuning, we fine-tune the model on a single downstream task and evaluate performance. When both pre-training and multitask fine-tuning are applied, pre-training is applied first, then multitask fine-tuning is applied before task-specific fine-tuning is applied.

4 Experiments

4.1 Data

For sentiment analysis, paraphrase detection, semantic textual similarity, and linguistic acceptability, the following datasets are used respectively: Stanford Sentiment Treebank (Socher et al., 2013), a subset of the Quora dataset, SemEval STS Benchmark (Agirre et al., 2013), and Corpus of Linguistic Acceptability (Warstadt et al., 2018).

4.2 Evaluation method

For the three tasks: sentiment analysis, paraphrase detection, and linguistic acceptability, the accuracy of the dev sets are used to evaluate performance. For semantic textual similarity, Pearson correlation of the true similarity values against the predicted similarity values for the dev sets is used to evaluate performance (Agirre et al., 2013).

4.3 Experimental details

For all experiments, the AdamW optimizer (Kingma and Ba, 2015) is applied with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-6}$. A learning rate of 1×10^{-3} is applied for LoRA fine-tuning and frozen BERT baseline, otherwise 1×10^{-5} is used. All experiments were run on a single Nvidia Tesla K40 12GB GPU. Batch sizes for all experiments were chosen to be the maximum multiple of 2 size which can fit in GPU memory, but capped at 64. For all fine-tuning experiments, 15 epochs were run, and the model with the highest dev set accuracy was taken. For all multitask fine-tuning experiments, 3 epochs of multitask fine-tuning were run. The weight matrices altered by LoRA are all weight matrices in the original BERT model.

For pre-training, all four tasks were fine-tuned with a 15-epoch and 30-epoch pre-trained BERT model parameter initialization. Upon observing the greatest improvement in performance for sentiment classification, experiments sweeping the number of epochs for pre-training from 1 through 30 were run for sentiment classification.

4.4 Results

Let SST be sentiment classification, PARA be paraphrase detection, STS be semantic textual similarity, and LIN be linguistic acceptability. Let FT be fine-tune and PLL be Per-Layer Learning rate.

4.4.1 Per-layer LR

Table 1: Baseline Acc/Corr Results

	SST	PARA	STS	LIN
Baseline Frozen	0.404	0.731	0.587	0.75
Baseline Fine-tune (FT)	0.513	**0.886**	**0.86**	0.844
Baseline FT and PLL	**0.518**	0.885	0.852	**0.85**

In table 1, We observe minor differences in performance from applying per-layer learning rates. The greatest performance improvement is in STS, which is expected because STS suffers from overfitting the most, with training accuracy reaching more than 0.9 while dev accuracy hovering around 0.52, and per-layer LR is a technique which aims to reduce overfitting by updating the more general-purpose lower layers less through a lower learning rate. Figure 1 demonstrates the importance of each layer towards performance, illustrated using the STS task.

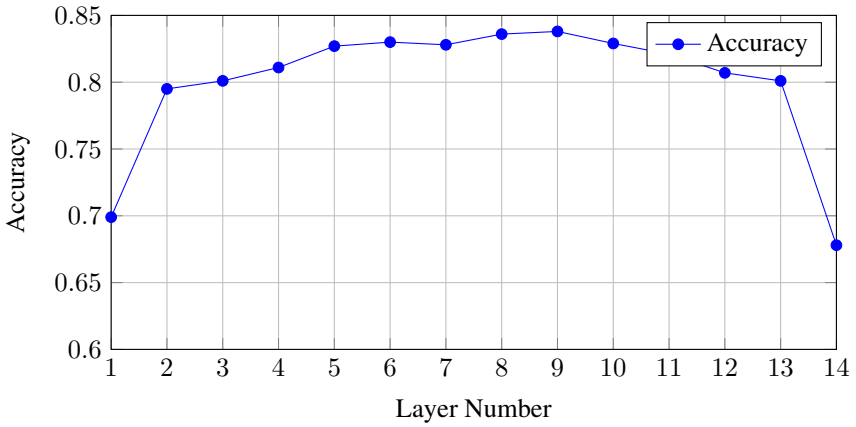

Figure 1: Accuracy vs. Layer Number

We observe that generally the lower layers contribute less to performance, the middle layers contribute the most, and surprisingly the last few layers contribute less than the middle layers, with the exception being the last layer, pooler layer. The embedding and pooler layer achieve significantly lower accuracies, implying that these layers not as important as the 12 attention layer for the transformer architecture. More research into different tasks and datasets would be desired to understand this result and the impact of each attention layer themselves.

4.4.2 LoRA

Table 2: Dev Set Accuracy/Correlation Results

	SST	PARA	STS	LIN
Baseline Frozen	0.404	0.731	0.587	0.75
Baseline Fine-tune (FT)	0.513	**0.886**	0.86	0.844
Baseline FT + PLL	0.518	0.885	0.852	0.85
FT + PLL + LoRA 1	0.513	0.873	0.859	0.861
FT + PLL + LoRA 2	0.502	0.882	0.865	0.861
FT + PLL + LoRA 4	**0.524**	0.882	**0.869**	**0.862**
FT + PLL + LoRA 8	0.506	0.884	0.862	0.861
FT + PLL + LoRA 4 and multitask	0.513	0.882	0.864	0.856
FT + PLL + task-specific pre-training 15-epoch + LoRA 4	0.484	0.858	0.749	0.761

In table 2, we observe that LoRA achieves similar performance as baseline FT and PLL. For LoRA with rank $r = 4$, we actually see minor performance improvements, which is surprising since LoRA typically should not perform better than full fine-tuning. This may be due to LoRA preventing some overfitting and learning a better representation which generalizes better to unseen data to low rank changes to weight matrices. We also observe that LoRA with multitask achieves similar performance, which makes sense since it's likely post fine-tuning, it would learn the same representation as without multitask fine-tuning applied.

Applying LoRA on a task-specific further pre-trained model, however, achieves much lower performance. This is somewhat unexpected, since individually, LoRA and pre-training improves performance slightly, however we see large performance drops when combining the two techniques. This may be due to further pre-training reducing the generality of BERT weights, and thus when applying low-rank changes to the weight matrices, the model does not generalize as well to different downstream tasks.

4.4.3 Further Pre-Training

Table 3: Dev Set Accuracy/Correlation Results

	SST	PARA	STS	LIN
Baseline Frozen	0.404	0.731	0.587	0.75
Baseline Fine-tune (FT)	0.513	0.886	0.86	0.844
Baseline FT + PLL	0.518	0.885	0.852	**0.85**
FT + PLL + task-specific pre-training 15-epoch	0.529	0.889	0.866	0.848
FT + PLL + all dataset pre-training 15-epoch	0.518	0.889	**0.879**	0.841
FT + PLL + task-specific pre-training 30-epoch	**0.535**	**0.895**	0.874	0.848
FT + PLL + all dataset pre-training 30-epoch	0.506	0.893	**0.879**	0.837

In table 3, we observe large performance improvements over the baseline from task-specific pre-training for SST, PARA, and STS. Additionally, we observe that pre-training for 30 epochs further improves upon performance compared to 15 epochs. This is expected, because past research has demonstrated that further pre-training improves performance the longer we train. We see the largest performance gain for SST, which makes sense because SST is a difficult problem, with state-of-the-art models achieving less than 0.6 accuracy. Further pre-training can improve the model's understanding of sentiment through prediction.

We observe performance improvements in all dataset pre-training for STS and similar performance for PARA. This makes sense because paraphrase detection and semantic textual similarity are tasks which are quite similar to each other, so pre-training on both datasets would benefit both tasks mutually. However, the dataset for PARA is much larger than STS, so PARA does not see much benefit whereas STS sees larger performance gains. We also observe that SST decreases in performance from all dataset pre-training. This also makes sense since none of the other tasks contain sentiment data, so the model does not improve its understanding of sentiment from pre-training on other daatasets. Moreover, pre-training is dominated by PARA and STS due to their task similarities and task getting equal weights during pre-training, which is not beneficial to SST and may adversely affect the model performance due to the model focusing on those tasks more.

However, we do not see much difference in performance for LIN. This may be because the task of linguistic acceptability is very similar to the MaskLM task which BERT was trained on during pre-training since BERT generally always used linguistically acceptable text. So, further pre-training on data similar to data which BERT was already trained on for many more epochs would not improve performance. Similarly, for all data pre-training, other tasks are not similar to linguistic acceptability, so they do not benefit performance.

Next, we show results for the impact of number of pre-training epochs on performance post fine-tuning for SST in Figure 2.

We observe the same trends as Sun et al. (2019), where the more pre-training steps run, the better performance the model achieves. If given more time and computational resources, we would continue further pre-training to see when performance plateaus.

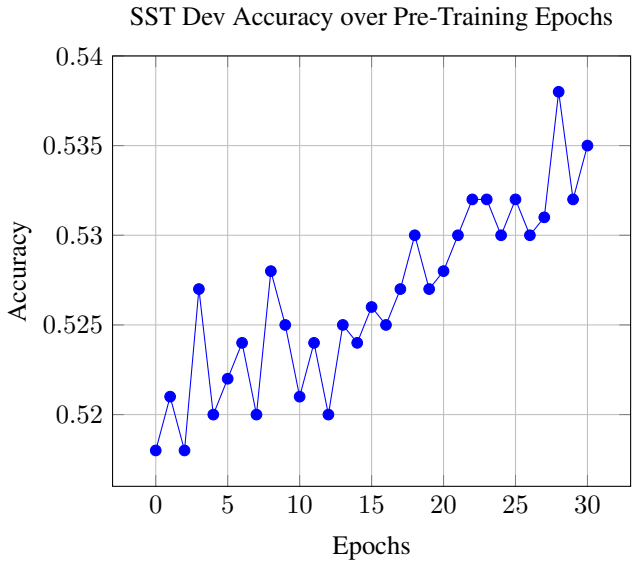

Figure 2: SST Dev Accuracy over Pre-Training Epochs

4.4.4 Multitask Fine-Tuning

Table 4: Dev Set Accuracy/Correlation Results

	SST	PARA	STS	LIN
Baseline Frozen	0.404	0.731	0.587	0.75
Baseline Fine-tune (FT)	0.513	0.886	0.86	0.844
Baseline FT + PLL	0.518	0.885	0.852	0.85
FT + PLL + multitask	0.523	0.889	0.866	**0.858**
FT + PLL + task-specific pre-training 15-epoch	0.529	0.889	0.866	0.848
FT + PLL + task-specific pre-training 15-epoch + multitask	0.527	0.89	0.882	0.848
FT + PLL + all dataset pre-training 15-epoch	0.518	0.889	0.879	0.841
FT + PLL + all dataset pre-training 15-epoch + multitask	0.514	0.892	0.877	0.835
FT + PLL + task-specific pre-training 30-epoch	0.535	**0.895**	0.874	0.848
FT + PLL + task-specific pre-training 30-epoch + multitask	**0.54**	0.893	**0.885**	0.85

In table 4, we observe improved performance from multitask fine-tuning when combined with task-specific pre-training for STS in particular. However, we observe mixed results for all other tasks. This result makes sense since STS may benefit from the features learned from the PARA task due to similarity in the two tasks. This is exciting, since this suggests that different domains of data or tasks can exhibit transfer learning properties and benefit the model's capabilities. However, PARA may not benefit much since its dataset is much larger and the desired features are already learned. We observe similar performance for SST and LIN, which may be attributed to the features and embeddings learned from other tasks not benefiting SST or LIN.

We observe that all dataset pre-training and multitask fine-tuning achieves similar performance. This may be because the objectives are similar between the two techniques, and the task diversity benefits are already reaped during pre-training, thus multitask fine-tuning not having much of an additional benefit in performance.

4.4.5 Test Results

	SST	PARA	STS	Overall
FT + PLL + task-specific pre-training 30-epoch + multitask	0.557	0.892	0.871	0.795

Table 5: Test Acc/Corr Scores

In table 5, we observe similar results as the results on the dev set. Although, accuracy on SST is higher than expected while STS accuracy is lower than expected. This may imply that the data distributions between the dev and train sets are slightly different for SST and STS, which can be verified and mitigated by experimenting on more data or different test and dev splits of the datasets given more research time.

5 Analysis

For sentiment analysis, we find 87% of the dev set outputs the model gets wrong are off by one. We also find the distribution of labels which the model gets wrong are concentrated towards the middle classes, which are sentences with more ambiguous sentiment. Manually inspecting these cases, we find it is reasonable for even humans to classify these incorrectly given the ambiguous nature of the task and the large number of sentiment classes. For example, the following sentence, "The heavy-handed film is almost laughable as a consequence," is labeled as 1 whereas the model classifies 0, and the sentence clearly has quite negative sentiment.

For paraphrase detection, we find the model gets simple examples incorrect, but also tricky examples incorrect. We also find the distribution of classes the model gets incorrect to be balanced. This suggests that there is some degree of over-fitting, but also the model may lack semantic understanding of some words or grammar structures. For example, the model may over-fit on some phrases such as subjects and verbs while giving less weight to other words when some words may completely change the meaning of a sentence as is in the case of the following sentence pair which the model incorrectly classifies as paraphrases: 'Why do people become rebels?', 'How someone become rebels?'

For semantic textual analysis, we observe both issues present in sentiment analysis and paraphrase detection. Analyzing the examples that the model deviates from the label by more than 1.0, we find that some examples may be ambiguous even for humans, while other examples demonstrate lack of understanding of certain words and phrases by the model.

For linguistic acceptability, we observe model tends to incorrectly label more sentences as acceptable. Analyzing the incorrect examples, we find the model lacks basic understanding of some words, grammar and vocabulary, which may be attributed the model size not being large enough to remember the semantic structures and features of all different types of words and phrases.

Through analyzing the outputs of the model, we find the model lacks understanding of some words, phrases, and semantic structures, suggesting the possibility of over-fitting, not a large enough model, or not enough pre-training. These issues may be mitigated by a larger model, trained on more data, and pre-trained for longer, which is consistent with the performance improvements we are currently seeing in research on large language models much larger than BERT, trained on much more data. It is exciting to see how far scaling can take us in the field of natural language processing.

6 Conclusion

We find pre-training and multitask fine-tuning to be very effective in improving the performance of BERT, and that LoRA performs well on general pre-trained weights but poorly on less general task-specific pre-trained weights. By achieving higher performance on difficult tasks by improving the model's understanding through learning by prediction using the MaskLM objective, utilizing diversity in data and tasks, and analyzing model outputs and where the model can be improved, the experiments in this paper supports the scaling laws exhibited by many recent LLM research which pursue scaling models parameter sizes, dataset diversity, and data modalities such as video, image, text, and audio. Limitations of this work include lack of scale in terms of dataset sizes, diversity, pre-training epochs, and modalities due to compute and time limitations. Further work may include research into the benefits of multiple modalities on pre-training and multitask fine-tuning at a smaller scale and exploring different methods of combining modalities to facilitate transfer learning between modalities.

References

Armen Aghajanyan, Lili Yu, Alexis Conneau, Wei-Ning Hsu, Karen Hambardzumyan, Susan Zhang, Stephen Roller, Naman Goyal, Omer Levy, and Luke Zettlemoyer. 2023. Scaling laws for generative mixed-modal language models.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus).

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

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

Luca Pasa. 2014. Pre-training of recurrent neural networks via linear autoencoders. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, page 3572–3580.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Neural network acceptability judgments. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuangjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference*, page 194–206, Online. Proceedings 18s.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments.