ExTraBERT: Exclusive Training for BERT Language Models

Stanford CS224N Default Project

Medhanie Irgau
Department of Computer Science
Stanford University
mirgau@stanford.edu

Chinmay Lalgudi
Department of Computer Science
Stanford University
clalgudi@stanford.edu

Abstract

BERT is a powerful generative model that can construct rich, contextualized word representations. In this paper, we investigate the ability of BERT to perform on three separate tasks: Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity. We first implement pre-trained BERT weights and fine-tune on each individual task, introducing an individual classifier to enhance performance across each of the three tasks. We then implement innovative feature engineering when predicting the tasks, and make use of different loss functions and extensions optimized for each individual individual task. We find that by optimizing individual performance on all three tasks, we obtained comparable and even higher accuracy and performance than a multitask classifier across all 3 tasks, obtaining an accuracy of 0.526 for SST, 0.880 for Paraphrase, and a pearson coefficient of 0.873 for STS.

1 Key Information to include

- Mentor: Hamza El Boudali
- External Collaborators (if you have any): None
- Sharing project: None

Both Medhanie and Chinmay contributed to the codebase as well as writing the paper. We both implemented the baseline, and Medhanie took charge of the different attention mechanism eespecially to boost the score of the SST task, while Chinmay applied different techniques such as Cosine Similarity and feature engineering to attempt to boost the scores of Paraphrase and STS. Overall, the work was equally distributed.

2 Introduction

The field of Natural Language Processing (NLP) has seen unprecedented advances over the last decade, connecting computational power with language. These developments allow machines to understand, interpret, and generate text similar to actual humans. Advances in this field have contributed to the rapidly evolving field of generative modeling, particularly with Large Language Models (LLMs) and other fields. NLP encompasses a wide range of applications, from automated translation and sentiment analysis to question-answering systems and beyond.

There are many challenges in NLP, with one of the greatest being inherent complexity of human languages, across cultures and dialects. This complexity is largely due to variability in context, ambiguity, and nuance, aspects of language that humans can pick up eventually over time, but machines have difficulty interpreting. A key moment in the evolution of NLP is the introduction of the transformer architecture, laying the foundation for computational models to achieve success interpreting a variety of language understanding tasks (Vaswani et al., 2017).

Among these, BERT (Bidirectional Encoder Representations from Transformers), harnesses the power of transformer architectures to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers (Devlin et al., 2019). BERT's ability to construct rich, contextualized word representations continues to be powerful in the field, yet despite BERT's success, the application of such models to specific NLP tasks requires not only the accurate interpretation of linguistics but also an understanding of subtler aspects of communication. We aim to study the ability to leverage a pre-trained BERT model on three separate tasks: Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity.

To address this problem of multitask performance, we start with pre-trained BERT weights and fine-tune on each individual task, introducing an individual classifier to enhance performance across each of the three tasks. We implement innovative feature engineering when predicting the tasks, and make use of different loss functions and extensions optimized for each individual task. In addition, to increase performance, we parallelized the input across multiple GPUs for efficient training. We find that by optimizing individual performance on all three tasks: sentiment analysis, paraphrase detection and STS, we obtained comparable and even higher accuracy and performance than a multitask classifier across all three tasks.

3 Related Work

The introduction of BERT was a significant milestone in NLP, allowing for the classification of multiple different language tasks, including the tasks of sentiment analysis, paraphrase detection, and SST. Leveraging transfer to achieve state-of-the-art performance across multiple NLP benchmarks, several variations of BERT have been developed to optimize efficiency and performance. In particular, RoBERTa extended BERT by training on a significantly larger dataset, removing the next-sentence prediction objective and dynamically changing the masking pattern applied to the training data (Liu et al., 2019). RoBERTA also demonstrated that hyperparameter tuning does in fact have a significant impact on the accuracy of a language model, emphasizing the need for an efficient approach in language model training.

Similarly, ALBERT presented a lite version that allowed for lower memory consumption and increasing the training speed of BERT, implementing a self-supervised loss that can understand inter-sentence coherence better, as well as parameter-sharing across layers. Multitask Learning (MTL) has emerged a promising approach for using BERT more efficiently by training a single model on multiple tasks simultaneously. The premise of this approach is that shared representations can improve generalization by leveraging commonalities across even marginally related tasks. Multi-task Deep Neural Network (MT-DNN) uses MTL to extend BERT via an additional layer to perform various tasks, displaying improved performance on benchmarks such as GLUE, SNLI, and SciTail. (Liu et al., 2019).

The effectiveness of BERT in multitask settings has also been explored in domain-specific applications. It was shown that the use of a unified BERT model outperformed task-specific models for specific tasks like joint entity and relation extraction in biomedical text (Peng et al., 2020), highlighting BERT's adaptability across tasks requiring deep domain knowledge. Adapting BERT for tasks like paraphrase detection and semantic textual similarity, as proposed in this paper, require a developed understanding of sentence structure and meaning. Models like Sentence-BERT have been able to generate specific sentence embeddings that can identify similar structured phrases within contexts (Reimers and Gurevych, 2019).

A key aspect of multitask learning is crafting adaptable loss functions, incorporating the balance of several different task-specific objectives. Dynamically balancing the loss contribution for each tasks highlights the importance of task selection and maximizing the value in joint training of one loss function. In summary, the development of BERT-based models and multitask classification algorithms have allowed for faster compute and training across several tasks, and in this paper we study the ability of BERT to perform across three objectives in particular, comparing multitask performance against individualized fine-tuning on separate tasks.

4 Approach

We are using BERT, a transformer-based model that generates contextual word representations, to perform the three downstream tasks of sentiment analysis, paraphrase detection, and semantic textual similarity. BERT first converts sentence input into tokens and then uses a trainable embedding layer across each token (Devlin et al., 2019). These learnable embeddings include token, segment, and position embeddings, which encode for contextual representations within text. BERT then makes use of 12 Encoder Transformer layers, consisting of multi-head attention, an additive and normalization layer, a feed-forward layer, and a final additive and normalization layer with residual connection (Vaswani et al., 2017). The BERT architecture also applies dropout to the output of sub-layers, a regularization technique to prevent overfitting of the training data.

After implementing BERT and applying it to a single sentiment analysis task over multiple datasets, we then implemented a multitask classifier for BERT that uses the weights of a pre-trained BERT model to predict sentiment analysis, paraphrase detection, and semantic textual similarity. Our baselines was then using these pretrained weights from the BERT model and finetuning both these weights and the classifier on the SST dataset (meant for sentiment analysis) (**?**). We then evaluated this model on all three tasks as the baseline, with the expectation that the model would have lower performance on the paraphrase and sentiment tasks. The accuracies and correlations for this baseline model on the 3 tasks are: 0.309, 0.38, and -.103, respectively, with a final score of 0.379.

This baseline model fine-tunes on simply sentiment classification, and was not able to generalize well to the other tasks. So initially, we implemented a single multitask classifier for all 3 tasks, noting the accuracy, correlation, and other model evaluation methods. We first used round robin curriculum to create an approach that is more generalizable across the three tasks. Within each batch, we iterate through the three SST, Paraphrase, and STS datasets, updating the individual parameters during each step. Our forward function returned the pooler output layer from the BERT embeddings, for all of our tasks. For SST, we apply dropout to the outputs of forward, and passit trhough a linear classifier layer to obtain the logits. For Paraphrase, we similarly apply dropout on the two outputs from the BERT model, which correspond to pairs of sentences. We then concatenate these and pass the combined embeddings into a linear classifier layer to obtain the logit. And for STS, we had a early identical to Paraphrase, except using a different linear classifier to obtain the logit. From these, we compute the forward and backward pass necessary for training, and within each batch we considered a combined loss, calculated to be the average of the three individual task losses. We used Cross-Entropy Loss for SST and Paraphrase and MSE Loss for STS. We finally took the gradient of combined loss with respect to our model parameters in order to update our weights.

However, we noticed that training each task separately led to subpar performance, and realized that the gradient directions of different were perhaps conflicting with one another — therefore, we incorporated gradient surgery to mitigate this. More specifically, we used the PCGrad implementation to ensure the gradients of the three tasks were projected onto the plane that was normal to each of the other task's gradients:

$$g_i = g_i - \frac{g_i - g_j}{||g_j||^2} \cdot g_j$$

In an attempt to improve the performance of the multitask classifier, we use the pretrained BERT weights and finetune a separate model on each dataset for the corresponding task. For the SST classification task, we take the ouputted embeddings from BERT and perform multiplicative attention, calculated as:

$$e_i = s^T h_i \in \mathbb{R} \text{ from } h_1, ..., h_N \in \mathbb{R}^{d_1} \text{ and } s \in \mathbb{R}^{d_2}$$

$$a = \sum_{i=1}^{N} \text{softmax(e)}_i h_i \in \mathbb{R}^{d_1}$$

Multiplicative attention has been found to be useful at enabling the model to attend to different parts of the input sequence with varying degrees of importance, helping incorporate relevant contextual information into its predictions, eventually leading to more accurate and context-aware outputs (Luong, Pham, and Manning 2015).

We pass this through a 3-layer feed-forward neural network, and utilize the output of this network in our predict sentiment function. Dropout is applied and a is passed through a linear sentiment classifier layer, outputting logits. For each class ranging from 0 to 4 we get a logit, and we calculate the Cross-Entropy Loss from these.

For the Paraphrase binary classification task, the forward pass is the same as prior, performing multiplicative attention and a feed-forward neural network. We perform the forward pass on the input ids and the masks (for each sentence), and first concatenate the two outputs together. Adding to this, we also concatenate the square of the sum of the two embeddings and the square of the difference of the two embeddings. From this feature engineering, we can learn not only the individual embeddings, but also the difference between the embeddings - a sense of graphical relativity between the embeddings. To ensure a non-negative sum and distance as well as symmetry between the embeddings, we apply this squared function to the sum and difference. We then concatenate the Cosine Similarity between the output embeddings as a more explicit similarity metric between the potential paraphrases, completing our engineering features. Finally, a linear classifier layer is applied to the total features, outputting logits. We then compute the Cross-Entropy Loss between the logits and the labels.

$$\text{Cosine Similarity(u, v)} = \frac{u \cdot v}{||u||||v||}$$

For the STS task, the forward pass consists of only multiplicative attention on the inputs ids and masks, with no feed-forward neural network, as determined empirically through testing. We then compute the cosine similarity between the two outputs, between 0 and 1. We scale this value by 5 because the STS output values for semantic text similarity are between 0 and 5, obtaining a single logit. We then compute the Mean Squared Error between the aforementioned logits and the true labels.

Due to the computational challenges these provided, we also implemented parallelizing the input for training across multiple GPUs. In the forward pass, the model is replicated on each GPU, parallelizing the input, and in the backwards pass, gradients from each replicated model are summed.

5 Experiments

5.1 Data

The data we will use is also outlined in the default paper spec. The sentiment analysis dataset is the Stanford Sentiment Treebank, consisting of 11,855 single sentence extracted from movie reviews, parsed with the Stanford parser and ultimately is made up of 215,514 annotated unique phrases (Socher et al., 2013). The actual dataset consists of sentence and id, and a corresponding sentiment value from negative to positive, 0 to 4 respectively. The paraphrase detection dataset is of a subset of the Quora dataset. Each example in this dataset consists of two sentences and whether they are semantic duplicates of eachother, corresponding to a 0 or 1. The STS dataset consists of two sentences and a value from 0 to 5 that assesses the similarity of the two phrases. In the baseline, we are simply finetuning our model and classifier on the SST dataset, saving this model and making predictions on the three tasks using those weights. However, we extend this to finetune on the corresponding dataset for each of the tasks - namely the SST data for sentiment analysis, Quota dataset for paraphrasing, and SemEval STS dataset for the semantic similarity task.

5.2 Evaluation method

We evaluate the model on the training dataset during finetuning, and a separate evaluation dataset after finetuning, for each task. For the sentiment analysis and paraphrase detection tasks, we evaluated using F1 score and accuracy, and for semantic similarity, we evaluated using Pearson Correlation Coefficient.

5.3 Experimental details

We ran the experiments using the finetuning option, which trains the BERT model and classifier. We used Weights and Biases to perform a "sweep" in order to intelligently choose hyperparameters,

namely batch size and learning rate. Essentially, we provided a configuration including an objective we wanted to maximize for each task, accuracy, correlation, etc. We then provided batch size values of 4, 8, 12 as well as learning rate bounds of 1e-4 and 1e-6. We configured a random method, so for each training count (we had 20 counts), a random learning rate within the proposed bounds as well as a random batch size from those proposed was selected. We did this for the SST and STS tasks. We did not do this for Paraphrase, and we did not optimize with respect to epochs due to time constraints, One of our sweeps graphs can be seen below.

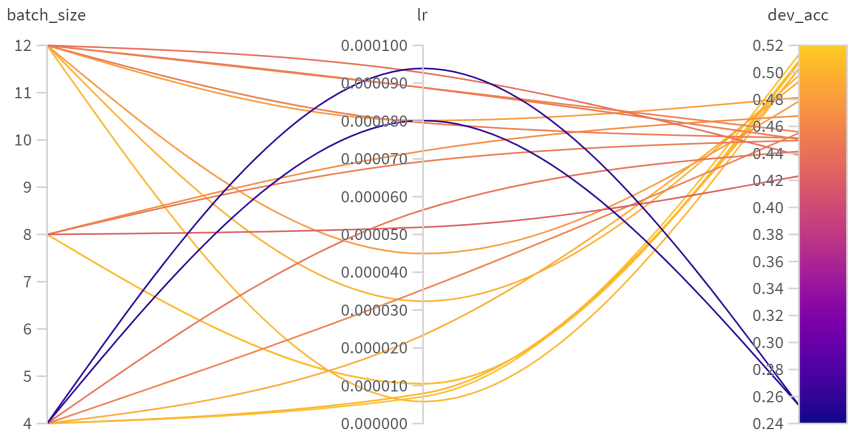

Figure 1: SST Hyperparameter W&B Sweeps

From these sweeps, we learned that the best (of 20) hyperparameter configurations were batch sizes of 12 and 12 as well as learning rates of 0.00000581232141486156 65 and 0.00001308967461164673 for SST and STS, respectively. Thus, we used these learning rates and batch sizes for our SST and STS runs with ten epochs.. We used ten epochs for Paraphrase as well with a batch size of 8 and learning rate of 1e-5. We also parallelized our code to work with Torch's DataParallel. We used two GPUs for training STS and SST and four GPUs for training Paraphrase. Training STS took ten minutes, SST eleven minutes, and Paraphrase almost five hours.

5.4 Results

When evaluating each of the models, we measure accuracy for the SST and Paraphrase tasks and Pearson Correlation for STS. In the following table, RR stands for Round-robin, GS stands for Gradient Surgery, SFT stands for Separate Fine-tuning, DPA stands for Dot Product Attention, MA stands for Multiplicative Attention, and CS stands for Cosine Similarity, and the following scores are on the dev set:

Model	SST	Paraphrase	STS
Baseline	0.309	0.380	-0.103
RR	0.482	0.686	0.380
RR + CS	0.484	0.686	0.625
RR + CS + GS	0.491	0.742	0.687
SFT	0.522	0.783	0.385
SFT + CS	0.514	0.785	0.641
SFT + DPA + CS	0.519	0.860	0.851
SFT + MA + CS	**0.526**	**0.880**	**0.873**

Table 1: Model accuracy and correlation on the dev set

We tested several models on the dev set as shown above, ranging from the baseline to modifications with round robin, and different types of attention with separate fine-tuning, as seen in Table 1. The SST, Paraphase, and STS increased drastically when using Round Robin, as this was able to generalize well among all 3 of the datasets. We found that using Cosine Similarity made a large difference

especially in the STS task, as concatenating the cosine similarity embeddings between the two output embeddings was able to encode differences between phrases in the vector space. Gradient Surgery allowed the model to generalize better across all the tasks, as expected, as the gradient directions were projected onto other tasks's gradients. However, the largest difference was when we decided to explore the possibility of specific fine-tuning, comparing these results to the multitask methods of round robin.

Adding Cosine Similarity to SFT allowed for a significant increase in the STS, as expected, due to similar reasons as above. We then decided to test multiple methods of attention in the forward pass for each of the individual models, before passing these into a feed-forward neural network, as described in a previous section. Dot-product Attention led to a large jump in the paraphrase accuracy and STS correlation, but ultimately Multiplicative Attention performed the best on all 3 of the tasks, obtaining an accuracy of 0.526 for SST, 0.880 for Paraphrase, and a pearson coefficient of 0.873 for STS. Multiplicative attention might be more effective than dot product attention int these multiple text semantic similarity and paraphrasing tasks because it allows for more fine-grained modulation of attention weights. By incorporating learnable parameters in the attention calculation, multiplicative attention is able capture complex relationships between words and phrases, resulting in improved semantic representation and better discrimination between paraphrases.

Our final submission of a separately fine-tuned model using multiplicative attention and cosine similarity on the test set performed as shown below:

Model	SST	Paraphrase	STS
SFT + MA + CS	**0.532**	**0.880**	**0.865**

Table 2: Best model accuracy and correlation on test set

Our final model was in **10th** place on the Default Project TEST Leaderboard at the time of submission, with comparatively high STS scores, likely due to the inclusion of Multiplicative Attention and Cosine Similarity.

6 Analysis

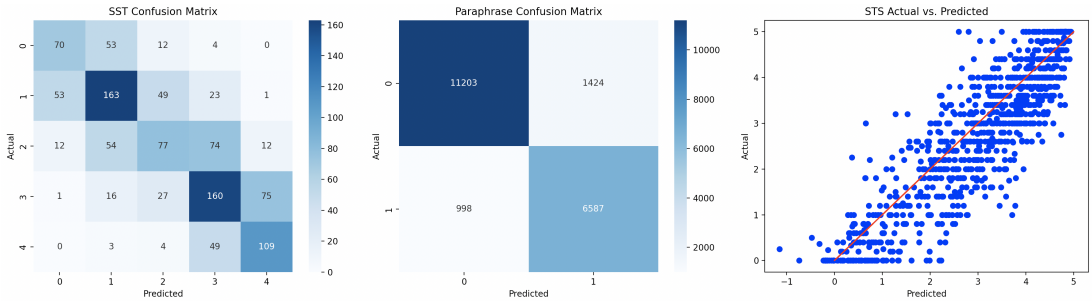

Figure 2: a) Confusion matrix for SST predicted and actual. b) Confusion matrix for Paraphrase predicted and actual. c) Scatterplot for STS predicted vs. actual.

We generate confusion matrices, heatmaps, and scatterplots for SST, Paraphrase, and STS, and we truly see the effectiveness of the model. As seen in Figure 2a, for SST, there appears to be highly accurate sentiments predicted for sentiments 1 and 3, which are moderate sentiments, whereas one would expect that the model would predict phrases with extreme sentiment with high accuracy. One potential reason for this could be due to the class imbalance present in the training dataset, as there are more target labels for 1 and 3 in the distribution.

In addition, there is an error band of wrong class predictions present around the sentiment ± 1, which is expected. From Figure 2b, the paraphrase confusion matrix shows a very high true positive and true negative values, with low false positive and false negative values, both quantitatively and visually. This error band also is present in the STS predictions, as seen in Figure 2c, with incorrect predictions occurring within this window for the regression task. One interesting feature to note is that some predictions are negative, which is theoretically not possible, but is a consequence of using a complex

algorithm for this regression task. This could be addressed in future steps to ensure that there is a cutoff or scaling within the task bounds.

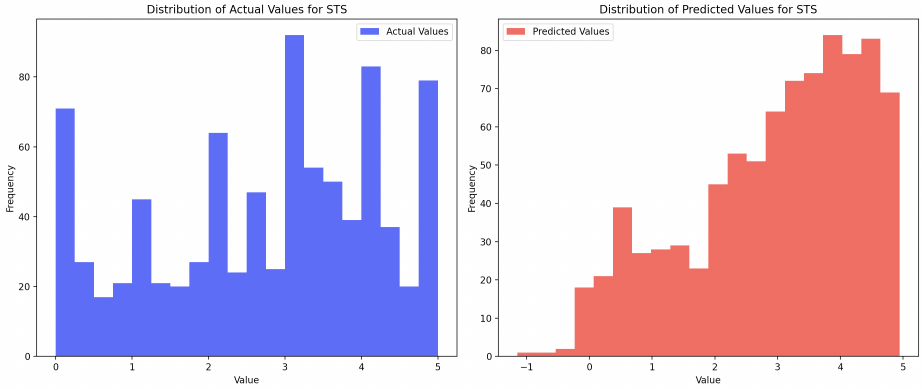

Figure 3: a) Histogram of distribution of actual values for STS on dev set. b) Histogram of distribution of predicted values for STS on dev set.

We see from Figure 3a that the distribution of actual values for STS is multimodal, containing several peaks at the integers from 0 to 5. This multimodality of the data could have potentially posed problems for the model as the data is supposed to be continuous yet is still almost discrete in nature. From Figure 3b, the predicted values appear to be skewed left, which could also be a result of this imbalance in the training data as mentioned above, and balancing out this data prior to developing a model could prove to be beneficial in the future.

Now, we evaluate our model qualitatively. Moreover, we will inspect our model's performance on the dev set to see where it succeeds and where it might go wrong. One example involves our model predicting an STS score of 1.98 for the sentences "Work into it slowly" and "It seems to work" when it should have been 0.0. BERT models are designed to understand the context of each word within a sentence. The model might be picking up on the contextual or thematic similarity between "Work into it slowly." and "It seems to work." Both sentences involve the concept of 'work' and a qualitative aspect ('slowly' vs. 'seems to work'), which might lead the model to assess them as somewhat similar, even if their literal intentions are different.

Delving into the architecture of BERT, its exemplary performance in predicting the sentiment score as 0 for the sentence "But it's too long and too convoluted and it ends in a muddle" underscores the model's advanced contextual comprehension. This proficiency is largely attributable to BERT's bidirectional design and its utilization of self-attention mechanisms. Unlike traditional models, BERT processes input data in both directions simultaneously, which empowers it to capture the intricate nuances of language context more effectively. Furthermore, the self-attention mechanism enables BERT to weigh the importance of each word in a sentence relative to others, facilitating a nuanced understanding of how terms like "too long" and "too convoluted" influence the overall sentiment in a specific context. This architectural sophistication allows BERT to discern the negative sentiment of the sentence by evaluating the contextual significance of each phrase, despite the potential neutrality or positivity of these words in different scenarios.

7 Conclusion

In this project, we investigated the ability and performance of BERT on three Natural Language Processing tasks — Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity. Experimenting with both multitask classifiers and fine-tuning separate models for each task, we discovered that the model with separate fine-tuning, multiplicative attention, and cosine similarity performed the best on all 3 tasks. While multitask classifiers allowed for faster compute and performance, accuracy in specific individualized tasks can still be improved, with separate models and classifiers outperforming on these three semantically challenging tasks.

In the future, it would be interesting to first of all normalize the distribution of the data before training the model, allowing for more equal representation of all classes in tasks. We would also

like to implement pre-training, utilizing a contrastive pre-training objective using comparison of adjacent sentences in unlabeled corpus data. Lastly, more hyper parameter tuning would be beneficial, allowing for deeper exploration of the hyper parameter space.

References

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Yifan Peng, Qingyu Chen, and Zhiyong Lu. 2020. An empirical study of multi-task learning on bert for biomedical text mining.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

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

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

A Appendix (optional)

If you wish, you can include an appendix, which should be part of the main PDF, and does not count towards the 6-8 page limit. Appendices can be useful to supply extra details, examples, figures, results, visualizations, etc. that you couldn't fit into the main paper. However, your grader *does not* have to read your appendix, and you should assume that you will be graded based on the content of the main part of your paper only.