My PAL BERT: Using Projected Attention Layers and Additional Fine-Tuning Strategies to Improve BERT's Performance on Downstream Tasks

Stanford CS224N Default Project

Colin Michael Sullivan
Department of Computer Science
Stanford University
cms548@stanford.edu

Abhishek Kumar
Department of Computer Science
Stanford University
a6kme@stanford.edu

Abstract

We examine different approaches to improve the performance of the Bidirectional Encoder Representations from Transformers (BERT) on three downstream tasks: sentiment analysis, paraphrase detection, and semantic text similarity (STS). Throughout our experiments, a variety of different fine-tuning strategies and advanced techniques were leveraged including implementing Projected Attention Layers (PALs), annealed sampling, multi-GPU training, unsupervised contrastive learning of sentence embeddings (SimCSE), fine-tuning on additional datasets, and adding a relational layer. We found that a combination of PALs, annealed sampling, and unsupervised SimCSE result in the largest improvements in overall system score.

1 Key Information to Include

TA mentor: Josh Singh. **External collaborators:** No. **External mentor:** No. **Sharing project:** No.
Colin Sullivan's Contributions: Implemented fine-tuning on additional datasets, Unsupervised SimCSE, LR warmup / decay, and the relational layer.
Abhishek Kumar's Contributions: Implemented PALs, annealed sampling, and multi-gpu training.
Common Contributions: Both Sullivan and Kumar contributed evenly to minBERT implementation and paper writing.

2 Introduction

Creating one language model that can accurately be applied to multiple downstream tasks is a major challenge in natural language processing (NLP). Early forms of Transfer Learning involved adopting a model trained on one domain to perform well in another. These models typically used only supervised learning datasets and resulted in systems that did not generalize well on multiple tasks. In recent years, pre-trained transformer models have become the dominant architecture for many NLP applications. Instead of randomly initializing weights and only training with labeled datasets, models are first pre-trained on a large language corpus and then fine-tuned on one or more downstream tasks. The pre-training process improves the model's ability to generalize, while the fine-tuning stage focuses the model on accomplishing a specific task. One of the first pre-trained transformer models was the Bidirectional Encoder Representations from Transformers (BERT)[1].

While the process of adapting pre-trained transformer models to multiple downstream tasks has proven effective, it does have certain challenges and limitations. Datasets for specific tasks may be limited, making training difficult. Aggressive fine-tuning may lead to the model overfitting and failing to generalize on the unseen data. Finally, a model may perform well on one downstream task, but fine-tuning on multiple tasks may degrade performance of the entire system.

We propose multiple methods to improve BERT's performance on three downstream tasks: sentiment analysis, paraphrase detection (para), and semantic text similarity (STS). Our resulting model improves the overall accuracy of our baseline minBert model by over 0.19 points to an overall test accuracy of 0.761 and overall dev accuracy of 0.770.

3 Related Work

Our work focuses on improving the performance of the pre-trained BERT model introduced by Devin et al. (2018) [1]. Below are multiple earlier works and contributions that inspired our experiments.

Parameter sharing is an effective way to fine-tune a pre-trained model for downstream tasks. In their MT-DNN study, Liu *et al.*, used task specific layers on top of the shared BERT layers to learn representations across multiple NLU (natural language understanding tasks) [5]. Rebuffi *et al.*, 2018 used Residual Adapter Modules in residual networks for multi task learning in computer vision [6]. In Stickland and Murray et. al (2019), projected attention layers were used to match performance of separately fine-tuned models on GLUE benchmark while using far fewer parameters [7]. Adding PALs to our BERT model may result in improved performance.

In Sun and Huang et al. (2020) [8], they investigate how multi-task fine-tuning on additional datasets can improve performance on text classification task. For their experiments, with the exception of the final classification layer, all tasks share the BERT layers and the embedding layer. The authors found that additional multi-task fine-tuning on certain datasets did improve overall accuracy. However some datasets, such as Yelp and AG's News, resulted in no change in performance. We take inspiration from their work and investigate whether adding new datasets to our training cycle will improve performance on certain downstream tasks.

Gao and Chen et al. 2022 propose a novel contrastive learning technique that they call contrastive learning of sentence embeddings (SimCSE) [2]. They introduce a supervised and unsupervised (SimCSE). Both approaches calculate the cosine similarity of semantic entailment and contradiction pairs to improve STS accuracy. They found that adding unsupervised SimCSE and supervised SimCSE to $BERT_{base}$ resulted in a 4.2% and 2.2% improvement respectively on STS compared to the previous best results. We investigate whether SimCSE can improve our model's STS performance.

Howard and Ruder et al. (2018) propose the Slanted Triangle Learning Rate (STLR) [3]. In STLR, the LR increases linearly at the beginning of training, and then decreases linearly forward the end of training. The goal here is prevent the model from overfitting during the early or late stages of training. Howard and Ruder found the STLR improved system performance when training with large datasets, but had less of an affect when the training datasets were smaller. We investigate adding STLR to our model to see if it can help the system generalize better.

4 Approach

4.1 Architecture Overview

Our approach focuses on implementing seven main modifications to BERT to improve system performance: 1. Projected Attention Layers (PALs), 2. Annealed Sampling, 3. Unsupervised SimCSE, 4. multi-GPU training, 5. fine-tuning on additional datasets, 6. LR warmup / decay, and 7. adding a relational layer. We coded all implementations below unless otherwise stated.

Projected Attention Layers (PALs): PALs focus on multi task learning by modifying the $BERT(.)$ function itself. Task specific parameters are added to each of the 12 layers of the BERTEncoder. See Figure A.1 for an illustration. Task specific function $TS(.)$ is added in parallel to BERTAttention inside each BERTLayer as given below.

$$h^{l+1} = LN(h^l + SA(h^l) + TS(h^l))$$

where l is the layer index of layer l. h^l is output of the layer l, $SA(h^l)$ is the BERTAttention output for layer $(l+1)$ and $TS(h^l)$ is the task specific output for layer $l+1$. The formulation for $TS(h)$ is given below.

$$TS(\mathbf{h}) = V^D g(V^E \mathbf{h})$$

Here, parameter h to the $TS(h)$ is hidden state of previous BERT Layer of dimension 768. V^E is a learnable linear projection of dimension $(204, 768)$. V^D projects back the output of $g(.)$ to size of hidden state, i.e 768. We have used $g(.)$ as a Multi Head attention. We take the $[CLS]$ token's embedding as the pooled output and use it for downstream tasks after projecting it through a linear layer [7]. While our PALs implementation is specific and unique to our BERT model, we did take inspiration from Stickland's open-source PALs code. [1]

Annealed Sampling: We experimented with three different training orders for our downstream tasks. First, we investigated a sequential training order that for each epoch, fine-tuned on the entire sentiment database, then paraphrase, and finally STS respectively. Second, we investigated a simple round-robin training order that for each epoch, trained on one batch from each of the three tasks until it reached the end of the smallest data set (in our case SST).

Finally, we investigated adding annealed sampling to our model. For each iteration, we selected a dataset to train with a probability determined by the size of the dataset over the sum of all datasets in the training corpora. After each epoch, we modify the probability to train more equally on each dataset. This can be modeled by the function below, where E is the total number of epochs[7].

$$\alpha = 1 - 0.8\frac{\epsilon - 1}{E - 1}$$

Unsupervised SimCSE: We experimented with adding unsupervised SimCSE to improve STS. Originally proposed by Gao *et al.*, 2022, the key idea behind Unsupervised SimCSE involves feeding the same input pair through the encoder twice [2]. By applying the standard dropout twice, we obtain two different encoding for the same input. Originally, we implemented Unsupervised SimCSE using cosine similarity and the loss function outline in Gao *et al.*, 2022. However after implementing PALs and using a new tokenizer that concatenates sentence pairs and separates them with the [SEP] token, we modified our SimCSE implementation to be used with linear layers and Cross Entropy Loss.

Fine-Tuning on Additional Datasets: We experimented with fine-tuning our BERT model on additional datasets including the Stanford Natural Language Inference (SNLI) and the CFIMDB corpara. Both of these datasets were integrated into our normal annealed sampling training loop.

We fine-tuned on additional datasets for para detection using the SNLI corpus, which consists of 570k English sentence pairs. Each sentence pair is labeled 0 (entailment), 1 (neutral), or 2 (contradiction). Our para detection model outputs a 1 if two sentences are paraphrases and a 0 if not. When parsing the SNLI dataset, we converted entailment labels to 1, contradiction labels to 0, and removed all neutral sentence pairs. We also fine-tuned on additional datasets for sentiment analysis using the CFIMDB dataset. This dataset contains 2,434 highly polarized movie reviews labeled with a 0 for a negative sentiment and a 1 for a positive sentiment. When parsing the CFIMDB dataset, we multiplied the labels by 4 as our sentiment analysis model outputs a score between 0 and 4, with 4 being a positive sentiment and 0 being a negative sentiment.

Multi-GPU Training: To accelerate training across our experiments, we trained our model in a Multi-GPU setup with 4 Nvidia T4 GPU. We used PyTorch's DistributedDataParallel (DDP) to parallelize our workload. The training data is partitioned into world size (number of GPUs - 4 in our case) and is loaded using DistributedSampler while creating the DataLoader. Each GPU processes its batch of data independently and computes the gradients. The gradients calculated after forward pass are averaged across all GPUs so that after backward pass, each model replica is updated consistently. Training and Validation was run using a Multi-GPU setup while the test was run using a single GPU.

LR Warmup / Decay: We experimented with LR warmup and decay. Specifically, we implemented the Slanted Triangle Learning Rate (STLR) as proposed by Howard and Ruder [3]. In STLR, the learning increases linearly from zero for the first 10% of total steps, then linearly decreases to 0 for the remainder of training.

[1] https://github.com/AsaCooperStickland/Bert-n-Pals

Relational Layer: We also experimented with exploiting commonality between similar down-stream tasks. In particular, we focused on the similarities between the STS and para detection tasks as both are comparing the similarity of sentence pairs. The main difference between these two tasks is that their outputs are different scales. We experimented with adding a linear relational layer for these tasks, where sentence embeddings were first passed through the relational layer before being passed to the task specific STS or para linear layer.

4.2 Baseline

For our BERT baseline, we implemented three simple architectures for each of our downstream tasks. The sentiment classification task trains with Cross Entropy loss and passes the BERT embeddings through a simple linear layer with five outputs, one corresponding to each of the five possible sentiment classifications. For para detection, we concatenate the two BERT input embeddings before passing them through a linear layer and training with Binary Cross Entropy (BCE) loss. Finally STS concatenates the two BERT input embeddings before passing them through a linear layer and training with Mean Squared Error (MSE). For our baseline fine-tuning training loop, we fine-tune the system in sequential order: first we fine-tune on the entire sentiment database, then paraphrase, and finally STS. Our baseline scores are listed in the table below:

Overall	Sentiment	Para	STS
0.616	0.458	0.740	0.303

Table 1: Baseline Scores

5 Experiments

5.1 Data

Our model utilizes five datasets: the Stanford Sentiment Treebank (SST), CFIMDB, Quora, SemEval STS Benchmark, and SNLI corpara.

For sentiment analysis, our model was fine-tuned and evaluated on the SST dataset. SST consists of 11,855 single sentences extracted from movie reviews where sentiment scores are integers ranging from 0 (negative) to 4 (positive). We also performed additional sentiment analysis fine-tuning using the CFIMDB dataset, which was described earlier in Section 4. For para detection, we fine-tuned and evaluated our model on the Quora dataset. The Quora corpus consists of 400,000 question pairs labeled 0 (sentence pairs are not paraphrases) and 1 (sentence pairs are paraphrases). We also performed additional paraphrase detection fine-tuning using the SNLI corpus which was also described earlier in Section 4. Finally, we fine-tuned and evaluated STS using the SemEval STS Benchmark. SemEval consists of 8,628 different sentence pairs labeled with floating point values from 0 (unrelated) to 5 (equivalent meaning). Table 2 provides the train, validation and test split of our tasks.

Dataset	Train	Validation	Test
SST	8544	1101	2210
STS	6040	863	1725
Paraphrase	283003	40429	80858

Table 2: Distribution of data across splits

5.2 Evaluation Method

For sentiment analysis and para detection, given the binary labels of the dataset we utilize accuracy as our metric. For STS, we calculate the Pearson correlation of the true similarity values against the predicted similarity values across the test dataset. Since we are focused on the best multitask classifier, we evaluate our models based on overall score, which is calculated using the following equation: (sentiment dev accuracy + (STS dev accuracy + 1) / 2 + paraphrase dev accuracy) / 3.

5.3 Experimental Details

Unless otherwise noted, all experiments had an LR of 2e-5, a batch size of 16, hidden dropout of 0.3, and were trained on 3 epochs with 1800 iterations per epoch. Additionally, all experiments used the AdamW Optimizer with default $\beta_s(\beta_1 = 0.9and\beta_2 = 0.999)$ and weight decay of 0.01. Finally, the full model was fine-tuned and no layers were frozen. Below are additional parameters for specific experiments:

BERT Base: We use the same BERT-base architecture as defined by Devin et al. (2018) [1]. This architecture has twelve attention heads, $d_{ff} = 3072$ and $d_m = 768$. The maximum sequence length was kept at 128. We collected the pooler_output (output of [CLS] token) for both the sentences and then used it to do classification and regression respectively.

PALs: Our PALs implementation uses the default configuration introduced by Stickland and Murray [7]. This configuration has 12 attention heads per layer and the size of each layer is d_s = 204.

Fine-Tuning on Additional Datasets: The batch size for the CMIMDB dataset is 8 due to the large size of each dataset example. When fine-tuning with annealed sampling, we reduced the probability of training on SNLI by a factor of 10 due to SNLI's large dataset size.

SimCSE: Our Unsupervised SimCSE uses a dropout of 0.1.

LR Warmup / Decay: We perform linear LR warmup for the first 10% of steps to 2e-5, then perform LR decay for the rest of training down to 0.

5.4 Results

Overall Results: Our best model, which we call BP1, included the PALs architecture, annealed sampling, and our modified unsupervised SimCSE for the STS classifier. On the Test Leaderboard, this BP1 model received an overall score of 0.761, SST test accuracy of 0.482, para test accuracy of 0.865, and STS test correlation of 0.872. BP1 also recieved a 0.770 overall score on the Dev Leaderboard. All tables and charts below reference dev data as it could be used to test all model variations. Table 3 shows the results of our experiments with different models. While all experiments resulted in dramatically better performance compared to our Baseline BERT implementation (model 0), additions to our best BP1 model resulted in no changes or even worse performance.

Model	Architecture	Additions	SST Dev	Para Dev	STS Dev	Overall
0	Baseline BERT	None	0.361	0.763	0.224	0.579
1	BERT + PALs	None	0.424	**0.863**	0.840	0.736
BP1	BERT + PALs	+ Annealed Sampling	0.490	0.861	0.866	0.761
3	BERT + PALs	+ Annealed Sampling + Unsupervised SimCSE	**0.516**	0.862	**0.866**	**0.770**
4	BERT + PALs	+ Annealed Sampling + Unsupervised SimCSE + Fine-Tuning Add. Datasets	0.446	0.859	0.865	0.746
5	BERT + PALs	+ Annealed Sampling + Unsupervised SimCSE + Fine-Tuning Add. Datasets + LR warmup / decay	0.466	0.860	0.860	0.752
6	BERT + PALs	+ Annealed Sampling + Unsupervised SimCSE + Fine-Tuning Add. Datasets + LR warmup / decay + RL Layer	0.442	0.861	0.853	0.743

Table 3: Model evaluation results

Annealed Sampling: Annealed sampling resulted in a major overall score improvement. When comparing model 1 (PALs without annealed sampling) and model 2 (PALs with annealed sampling), you can see that SST, STS, and the overall score improved when annealed sampling was added. In particular, SST correlation saw the largest improvement jumping from 0.424 to 0.490, while STS saw a modest 0.02 point increase as well. Para detection remained mostly unchanged. These results are inline with what we expected. As with Stickland and Murray, annealed sampling improved their overall score by allowing the system to more evenly train on all datasets [7]. Figure 1 below compares the Para, SST, and STS training loss for model 1 (PALs with annealed sampling in pink) and model 2 (PALs without annealed sampling in blue). The SST and STS train loss charts below clearly show how annealed sampling helped lower their respective train loss by allowing the system to train more evenly on all datasets regardless of their size.

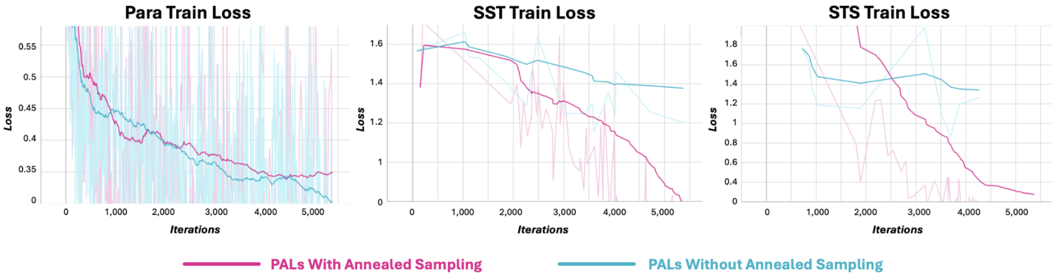

Figure 1: Training loss for each downstream task between model 1 (PALs with Annealed Sampling in pink) and model 2 (PALs without annealed sampling in blue)

Fine-Tuning on Additional Datasets: Performing fine-tuning on the SNLI and CMIMDB datasets decreased the overall score. Table 3 shows that when comparing our BP1 model (without fine-tuning on additional datasets) and model 4 (with fine-tuning on additional datasets) SST correlation dropped from 0.516 to 0.446 while para and STS remained relatively unchanged. These results are not what we expected. By fine-tuning on additional paraphrase and sentiment data, we expected these additional datasets to improve system performance. In Figure 2 below, we can see the para train loss and SST train loss for our BP1 model (shown in pink as "No Fine-Tuning on Additional Datasets") and model 4 that fine-tunes on additional datasets (shown in green). While fine-tuning on additional datasets does not seem to have a major effect on the para train loss, it does appear to have caused overfitting for SST as its loss was lower and its correlation score was also lower.

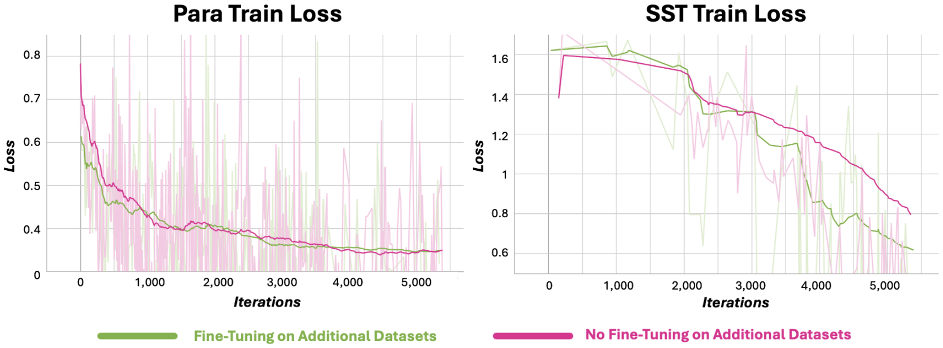

Figure 2: Para and SST Train Loss between BP1 model (shown in pink) and model 4 that fine-tunes with additional datasets (shown in green)

LR Warmup / Decay: LR warmup / decay resulted in no major changes to the overall score. When using Table 3 to compare model 4 (no LR warmup /decay) with model 5 (LR warmup / decay), we can see that the scores remained relatively consistent across the board. These results are not what we expected. Based on Howard et al. (2018)[4], we believed LR warmup / decay would prevent overfitting of early and late data in the training cycle, and lead to an improved multitask classifier. In Figure 3 below, we show the training loss for para detection where the blue line represents model 4 with no LR warmup / decay and the orange line represents model 5 with LR warmup / decay. As you can see in the figure, LR warmup worked as expected where the train loss was higher for the first 10% of steps as the LR increased linearly. Ultimately though, model 4 without LR warmup / decay and model 5 with LR warmup / decay converged to the same loss value.

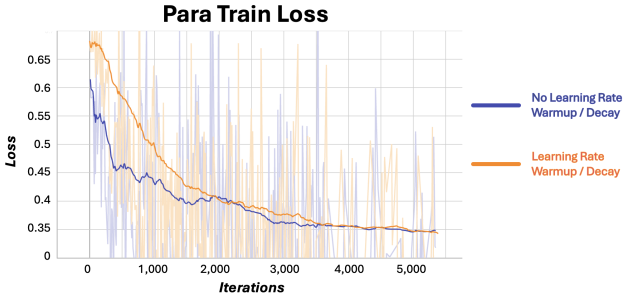

Figure 3: Para train loss between model 4 (No LR warmup / decay in blue) and model 5 (LR warmup / decay in orange)

6 Analysis

Projected Attention Layers: PALs resulted in massive improvements on the overall score and all 3 downstream tasks. We believe PALs improved the system score across the board because it adds additional task specific parameters in the model which are trained in addition to the different layers of BERT. Adding task specific parameters early on enables the parameters to learn richer representation of signals, as compared to adding them after getting [CLS] embedding from BERT.

Annealed Sampling: Annealed sampling resulted in major overall score improvements. This addition prevented overfitting on larger datasets and enabled us to sample tasks more uniformly during the last phases of our training. This helped us to reduce the risk of interference, where training heavily on a particular subset of tasks impacts the performance of other tasks. The Quora dataset used for para detection has over 400,000 sentence pairs, while the datasets used for STS and sentiment classification have closer to 8,000 examples each. Annealed sampling helped eliminate the issue where we were training too heavily on paraphrase detection examples while not training enough on STS and sentiment.

Fine-Tuning on Additional Datasets: We performed fine-tuning on additional datasets for sentiment and para, which actually lowered the score of SST and left para relatively unchanged. SST correlation may have decreased because the CMIMDB dataset only has extremely negative (0) and positive (4) sentiments and nothing in between. Figure 4 shows the data distribution for train and dev SST data. Since the distribution contains many sentiment labels between 0 and 4, the CMIMDB dataset may have caused the system to skew towards the 0 and 4 extremes when predicting. The SNLI dataset that we used for para detection had no change on system performance. We believe this may be because the existing Quora dataset used to train paraphrase detection is already very large. As a result, adding more data to train does not seem necessary.

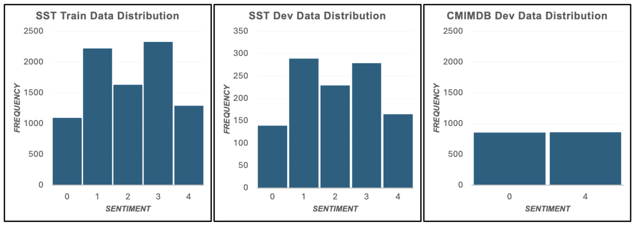

Figure 4: Data distribution for SST Train, SST Dev, and CMIMDB

Unsupervised SimCSE: Surprisingly, unsupervised SimCSE did not improve performance for STS which is the task it was implemented for. Instead, it resulted in an SST improvement from 0.490 to 0.516. It appears Unsupervised SimCSE allowed the model to generalize better and not overfit to a specific task.

Relational Layer: Our relational layer for the similar paraphrase and STS tasks resulted in no noticeable change in system score. This may be because the relational layer we constructed was a simple linear layer. Further experiments are required to determine whether a more complex neural layer could improve performance.

LR Warmup / Decay: LR Warmup / Decay did not affect overall performance. This may be because in Howard et al. (2018) [4], the authors noted that this feature only had a noticable affect on models being trained with large datasets. It is possible our dataset was not large enough for LR Warmup / Decay to have an effect.

7 Conclusion

We have detailed our modifications to BERT to fine-tune it on sentiment analysis, para detection, and STS. These modifications include 1. PALs, 2. Annealed Sampling 3. Unsupervised SimCSE, 4. multi-GPU training, 5. Fine-tuning on additional datasets, 6. LR warmup / decay, and 7. adding a relational layer. PALs and annealed sampling resulted in the largest improvement to overall accuracy, enabling a richer representation of signals and more even training on all data. Our best model, BP1, included BERT with PALs, annealed sampling, and a modified Unsupervised SimCSE. This final BP1 model improved on our base minBert implementation by an overall accuracy of 0.19 points. Future experiments could investigate whether additional pre-training on in domain datasets, along with adding a more complex relational layer could result in further improvements.

8 Ethics Statement

Utilizing minBERT for sentiment analysis, paraphrase detection, and STS tasks can pose multiple ethical challenges and societal risks. Here, we will outline one ethical challenge / society risk for each of the three downstream tasks. Sentiment analysis tools can be used to identify hostile or harmful posts on social media sites. If the sentiment analysis tool has bias, it may incorrectly identify posts as being harmful when they are not. More specifically, the sentiment analysis tool may be biased against a particular religion, culture, or viewpoint and may inaccurately target these posts as negative, while favoring posts about other religions, cultures, or views. For para detection, these systems could be used by governments or corporations for content censorship. Paraphrase detection tools could be used to search through large databases of files and remove data with specific content, allowing for censorship and the ability to control what the public sees. For STS, these tools can be used to identify plagiarism in work. If the STS tool is unreliable, it could inaccurately state that a text was plagiarized when it was not, unfairly harming the submitter's reputation.

A possible fix for the sentiment analysis tool being biased against certain views would be to ensure that the system is trained on a large, diverse dataset. A dataset that encompasses various cultures, religious beliefs, and ideas will help remove potential biases in the system against a particular group of people. For the paraphrase detection issue where individuals could use the system for content censorship, the system could be implemented with a limit on the number of inputs / searches it performs each day, preventing individuals from using it to quickly censor a large amount of content in a single day. Additionally, adding transparency features to the system, such as one that tracks all the system's inputs and outputs, could help inform individuals and the public how the system is being used. Finally, for the STS tool incorrectly identifying plagiarism, a transparency feature that shows users why the system arrived at its score that determined whether the text was plagiarized, could help users understand if the analysis was correct or incorrect. Additionally, the system could link users to the text that it believes is being plagiarized, highlighting areas that are similar.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[2] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821.

[3] Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

[4] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

[5] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504.

[6] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. *CoRR*, abs/1705.08045.

[7] Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR.

[8] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*, pages 194–206. Springer.

A Appendix

A.1

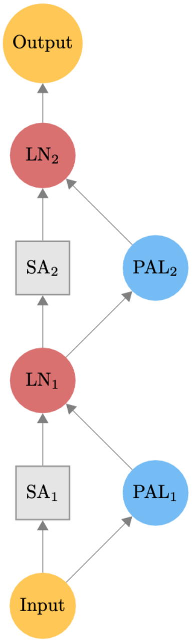

Figure 5: Training loss for each downstream task between model 1 (PALs with Annealed Sampling) and model 2 (PALs without annealed sampling)