Mastering minBERT: A True Balancing Act

Stanford CS224N Default Project

Emma Escandon
Department of Computer Science
Stanford University
emmaee@stanford.edu

Alejandro Rivas
Department of Computer Science
Stanford University
alex502@stanford.edu

Daniela Uribe
Department of Computer Science
Stanford University
duribe21@stanford.edu

Abstract

The performance of a baseline minBERT model on sentence-level tasks such as sentiment analysis, paraphrase detection, and semantic textual similarity (STS) can be enhanced through fine-tuning techniques. Our project investigates techniques that can improve the baseline minBERT model across all three tasks, both globally and independently. We experiment with neural network architectures, optimizers, regularization, additional datasets, ensembling, and different training techniques. Our best model achieved a score of 0.796 on the DEV set and 0.786 on the test set. From our results, we can see that the cross-encode architecture and hybrid training technique provide the largest score increase, complemented by smaller increases caused by SMART, ensembling, and additional datasets.

1 Key Information to include

Mentor: Arvind Mahankali, **External Collaborators**: N/A, **Sharing project**: N/A, **Contributions**: Emma Escandon: Writing abstract, introduction, SMART related work, conclusion, helped with SMART, ran models using Modal, helped create poster

Alejandro Rivas:

Daniela Uribe: Covering related work and experiments related to round-robin scheduling for task training, describing main datasets and a few extension datasets, implementing and experimenting with learning rate and optimizer fine tuning as well as round-robin scheduling variations to improve model performance, describing the model performance in the analysis, writing the ethic statement

2 Introduction

In recent years, the landscape of natural language processing (NLP) has been significantly transformed by advancements in pre-trained language models. Among these, BERT (Bidirectional Encoder Representation from Transformers) (Devlin et al. (2019) has emerged as a foundational model, demonstrating good performance across a wide range of NLP tasks. In this project, we use a smaller and more efficient variant, minBERT, due to the substantial computational requirements of the full BERT model.

This project explores the application of fine-tuning methods and embedding adjustments to enhance minBERT's performance on the following three tasks: sentiment analysis, paraphrase detection, and semantic textual similarity (STS). We aim to find the right balance between using as much information as possible from the training data while alleviating the vulnerability of fine-tuned models

to overfitting. This may be challenging since these two notions are at odds with each other, yet we strive to achieve significant improvements over the baseline minBERT model.

3 Related Work

SMART The purpose of the SMART (Jiang et al. (2020)) framework is to improve generalizable performance through the robust and efficient fine-tuning of pre-trained models. The new learning framework consists of two parts: Smoothness-inducing Adversarial Regularization technique, which effectively controls the high complexity of the model, and a class of Bregman Proximal Point Optimization methods, which prevent aggressive updating. More specifically, the authors applied SMART to BERT and RoBERTa pre-trained models and demonstrated that $SMART_{BERT}$ and $SMART_{RoBERTa}$ outperformed the strong baseline BERT and RoBERTa pre-trained models across all 8 General Language Understanding Evaluation (GLUE) benchmark tasks.

Sentence-BERT Sentence-pair regression tasks, such as STS, have been advanced by BERT however these models process both sentences together which can lead to substantial computational overhead. Sentence-BERT (SBERT) (Reimers and Gurevych (2019)) addresses this by modifying BERT with siamese and triplet network structures to generate semantically meaningful embeddings compared using cosine similarity, reducing similarity search time while maintaining accuracy. SBERT's effectiveness on STS and transfer learning tasks highlights its impact and why we decided to use this technique in our project.

Round-Robin Scheduling for Task Training Research regarding multi-task fine-tuning for BERT often points to the order in which the pre-trained model is fine-tuned for the various target tasks. Training on all of the data for each task sequentially often leads to a less robust embedding due to the lack of diversity in the training over time. Employing scheduling techniques such as round-robin scheduling to vary the order in which the model is trained for various tasks is one of many suggested methods in research behind curriculum training. These methods aim to optimize data ordering and selection to reduce training time while achieving the same performance goal (Asa Cooper Stickland (2019)). We were inspired by the promising research behind curriculum training to experiment with both sequential and round-robin batch scheduling as extensions in our project.

4 Approach

4.1 Baseline: minBERT Implementation with AdamW Optimizer

Per the project handout, we implement minBERT based on the architecture described in the paper presenting BERT(Devlin et al. (2019)). Within the BERT model each attention is calculated using the equation given in the paper, "Attention Is All You Need"(Vaswani et al. (2017)). We also implemented the AdamW(Loshchilov and Hutter (2019)) optimizer using the efficient version of bias correction given in "Adam: A Method for Stochastic Optimization"(Kingma and Ba (2017)). For our baseline, we used BERT embeddings in all tasks followed by a dropout layer. For sentiment, we used a softmax classifier. For paraphrase and STS we used the cosine similarity between the embeddings of the two different sentences (bi-encoder). Our baseline yielded a DEV score of 0.6283, with 0.5080 accuracy in sentiment, 0.6920 accuracy in paraphrase detection and a correlation of 0.3700 for STS.

4.2 Neural Network Architecture and Model Configuration

We retained the original architecture for sentiment classification, but we experimented with a cross-encoder architecture based on the `[CLS]` token only (Thakur et al. (2021)), which outperformed our previous one BERT embedding for both the paraphrase and STS tasks.

For the loss functions, we used cross entropy loss for sentiment classification, binary cross entropy for paraphrase detection and mean squared error for STS. We also kept the WAdam optimizer, although we did experiment with other optimizers like RAdam, NAdam and stochastic gradient descent.

4.3 Fine-tune Training Techniques

Given that the three datasets we used to train our model were imbalanced in size (283,011 examples for paraphrase, 6,040 for STS, and 8,544 for sentiment classification), we experimented with various

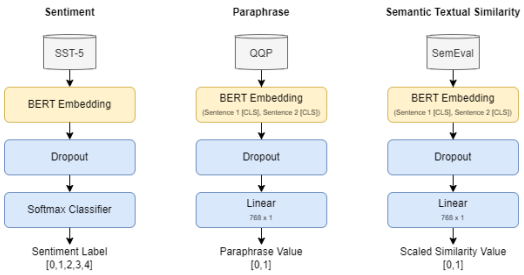

Figure 1: Cross-encoder architecture for paraphrase and STS tasks. Maximized BERT embeddings by concatenating the sentences before the BERT layer.

training techniques to balance the training process without undersampling the QQP dataset or oversampling the SemEval or SST-5 datasets. In most techniques, we prioritized the SST-5 dataset due to its lower accuracy compared to the other tasks. We also observed that increasing the training for either the paraphrase or STS tasks tended to implicitly improve the accuracy of the other.

Full sequential training. Training is conducted over the entire datasets in each epoch, with the order determined by the number of examples. We first trained on QQP, followed by STS, and then SST. After each dataset iteration, an optimizer step is taken instead of summing the losses at the end. Given that the QQP dataset is much larger than the other two, we also configured an option to repeat the SST and STS datasets multiple times within a single epoch.

Step-based training. We define a step size as a parameter and sample the datasets accordingly. If the step size exceeds the number of batches in a dataset, the sampling cycles through the batches in the same order. After each epoch, the data is randomly shuffled. In this approach, we experimented with two methods: a combined-loss approach, where a single optimizer step is used for all three losses combined, and a round-robin approach, where an optimizer step is taken for each batch individually.

Hybrid training. We first run one epoch of full sequential training, staring with QQP, followed by STS and SST. We then do step-based training. This way, we can make use of all the data available, while balancing future fine-tuning.

Single-task Fine-tuning. For some tasks, we noticed that further fine tuning using only the last linear layers improved the performance. While this is prone to overfit the data, we noticed a score in the dev accuracies as well.

4.4 Smoothness Inducing Adversarial Regularization - SMART Loss

We noticed that our training accuracies were much higher than our dev accuracies, which could indicate overfitting. As mentioned before, one of the methods presented in the SMART paper (Jiang et al. (2020)) to help mitigate this issue is Smoothness-inducing Adversarial Regularization. This regularization method leverages the principle of local Lipschitz continuity to maintain stability in the output of the function f when small perturbations (defined by the l_p norm ϵ) are introduced to the input data. The optimization problem is defined as

$$\min_{\theta} \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda \mathcal{R}(\theta)$$

where $\mathcal{L}$ is the single task's loss function, $\lambda > 0$ is an adjustable regularization weight and $\mathcal{R}$ is defined as

$$\mathcal{R} = \frac{1}{n} \sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} l(f(\tilde{x}_i, \theta), f(x_i, \theta))$$

where $\tilde{x}_i$ represents the perturbation in the embedding. As recommended by the original paper, we used symmetric KL-divergence as the loss function for sentiment classification and MSE loss for paraphrase and STS.

We made use of this technique using a pre-implemented pytorch library, smart-pytorch (AI (2022)). The tunable parameters are the number of optimization steps to find noise (num_steps, default = 1), step size to improve noise (step_size, default = 1e-3) , noise norm constraint (epsilon, default = 1e-6),

and noise norm constraint (noise_var, default = 1e-6). This regularization value is multiplied by a weight (λ) and applied to the loss. We used this default values, while varying λ.

4.5 Additional Datasets

We incorporated additional datasets in two different ways. First, we performed task-agnostic fine-tuning of the BERT parameters, prior to our multitask fine-tuning, using Natural Language Inference (MultiNLI and SNLI) datasets, creating $BERT_{NLI}$.

Secondly, we extended the training datasets for the sentiment task using 9,000 samples extracted from the Amazon Movie Review dataset and 9,000 from the Yelp dataset (separately), with an even number of samples per label. The training corpus for SST increased from 8,544 to 17,544. We did the same for STS, using the SICK dataset to further extend the training corpus from 6,040 to 11,040.

4.6 Ensembling

After doing thorough experimentation, we created an ensemble of 3 models, our top 2 overall models and our NLI-based model (to include balance out predictions). We used a bagging approach. For STS, we averaged all predictions. For sentiment prediction and paraphrase detection, we experimented between using majority vote and median to determine the final prediction.

5 Experiments

5.1 Data

5.1.1 Main Datasets

SST-5: Contains 215,154 phrases extracted from movie reviews with labels "negative," "somewhat negative," "neutral," "somewhat positive," or "positive." This dataset, generated from 11,855 sentences using the Stanford parser. This is the main dataset for sentiment classification task.

Quora Question Pairs (QQP): Contains 404,298 question pairs with binary labels indicating if the question pairs are paraphrased versions of each other. This is the main dataset for paraphrase detection task.

SemEval STS: Contains 8,628 sentence pairs with similarity labels from 0 (unrelated) to 5 (equivalent meaning). We use the Pearson correlation of the true similarity values against the predicted values to evaluate the performance of the model. This is the main dataset for semantic textual similarity task.

5.1.2 Additional Datasets for BERT Fine-tuning

SNLI: Contains 570,152 sentence pairs with labels "Entailment," "Neutral," or "Contradiction". This dataset provides information to understand if a sentence has a contradictory, consistent, or neutral relationship with its premise at a novel scale. This dataset will be useful to expand on the model's understanding of natural language and contextualization, potentially leading to a more robust semantic representation. We used the gold label, raw sentences and pair IDs, without any further pre-processing.

MultiNLI: Contains 433,000 sentence pairs with labels "Entailment," "Neutral," or "Contradiction". This dataset expands on SNLI, addressing the fact that SNLI is only derived from image captions. MultiNLI includes a wide diversity of written and spoken speech to improve a model's to capture the complexity of the English language. We used the gold label, raw sentences and pair IDs, without any further pre-processing.

5.1.3 Extension Datasets for Single-task Fine-tuning

Sentences Involving Compositional Knowledge (SICK): Built from the 8K ImageFlickr data set and the SemEval 2012 STS MSR-Video Description data set (Marelli et al. (2014)). It's composed of sentences that describe the same picture or video, thus being near paraphrases. The label for this dataset is a score between 0 and 5, similar to the SemEval STS dataset. We created an STS-extended dataset, composed of the original SemEval's training dataset and SICK dataset.

Amazon Movie Reviews (AMR): Dataset with 8 million movie reviews from Amazon, collected from 1997 through 2012 (Leskovec and Krevl (2014)). The label for this dataset is the rating that a user provided, from 1 to 5 stars. We extended our sentiment's training dataset with a sample from Amazon Movie Reviews. We created a parser that extracts reviews using the following criteria. Reviews must be between 20 and 512 characters long, the review ID and review itself should be unique. We transformed the movie rating into a score between 0 and 4 to match the SST dataset labels. Using this criteria, we extracted a total of 10,000 samples from the 8M-rows dataset, balancing out the labels (2,000 samples from each label).

Yelp Reviews Corpus: Dataset with 7 million reviews collected from metropolitan areas in the United States and Canada (Yelp (2023)). The reviews are matched to a 1 to 5 star rating, which we adapted to match the 5 categories of the sentiment classification labels. To prepare the dataset for sentiment classification single-task fine-tuning, we sampled 20,000 for each of the sentiment categories, ensuring that each review met the max embedding size for BERT of 512 tokens.

5.2 Evaluation method

We are evaluating our model's performance using accuracy for sentiment and paraphrase detection, and Pearson correlation for semantic textual similarity. To measure the multitask score, we used the following:

$$\texttt{Score} = \frac{\texttt{SST Acc.}}{3} + \frac{\texttt{Paraphrase Acc.}}{3} + \frac{\texttt{STS Corr.} + 1}{6}$$

Comparisons will be made against existing scores on each dataset to gauge improvement. As we implemented extensions on the original model, we kept track of the change in accuracies and whether or not the changes implemented have made significant improvement. We also submitted to the dev and test leaderboard and tracked improvement there as well.

5.3 Experimental details

We used the same hyper-parameter configuration for our baseline as well as our further experimentation. Learning rate of $1e-5$, batch size of 8 for all datasets (unless specified otherwise), 10 total epochs, using the values from the epoch with the highest score. We also used a dropout rate of 10% dropout for BERT embeddings, and a 10% dropout rate for single task embeddings.

5.4 Results

5.4.1 Neural Network Architectures and Optimizers

Table 1: DEV scores for different architectures and optimizers

Architecture and optimizer	**SST-5** Accuracy	**QQP** Accuracy	**STS** Correlation	**Overall** Score
Baseline Bi-Encoder + WAdam	0.5080	0.6920	0.3700	0.6283
Cross-Encoder + WAdam	**0.5168**	**0.7910**	0.8271	**0.7404**
Cross-Encoder + SGD	0.2920	0.6280	0.2830	0.5206
Cross-Encoder + RAdam	0.4970	0.7720	0.8150	0.7254
Cross-Encoder + NAdam	0.4950	0.7880	**0.8320**	0.7331

From our experimentation, we can see that there is a huge improvement by switching to a cross-encoder architecture. This makes sense because the BERT layers have more adjustable parameters that can help learning the underlying structure of the two sentences, rather than trying to come up with comparable embeddings through cosine similarity. We also decided to test a couple of optimizers to see how they would perform. Overall, WAdam had the highest performance. For all the upcoming runs, we froze used the cross-encoder architecture for paraphrase detection and STS with WAdam optimizer.

5.4.2 Fine-tuning Training Techniques

For all our runs, we used 10% dropout rate. Full sequential and round robin training used a batch size of 64 for SST and 8 for the other two. The rest used a batch size of 8 for all datasets. Hybrid uses 755 steps and 1069 steps because that's the total number of iterations required to loop through the entire STS and SST datasets, respectively. All the following models were run on a NVIDIA 4080 GPU.

Table 2: DEV scores for different training techniques.

Fine-tuning Training Technique	**SST-5** Accuracy	**QQP** Accuracy	**STS** Correlation	**Overall** Score	**Training Time** Per Epoch (hh:mm)
Full Sequential (10,1,5)	0.5032	0.8793	0.8597	0.7708	00:40
Full Sequential (6,1,3)	0.4959	0.8934	0.8660	0.7740	00:35
Full Round Robin (10,1,5)	0.5120	0.8730	0.8560	0.7711	00:40
Full Round Robin (6,1,3)	0.5005	0.8885	0.8473	0.7709	00:35
Step-based (3000 steps, combined)	**0.5168**	0.8657	0.8532	0.7697	**00:04**
Step-based (3000 steps, round-robin)	0.5020	0.8520	0.8520	0.7601	00:07
Hybrid (755 steps, combined)	0.5032	**0.8952**	**0.8702**	**0.7778**	00:32 + 0:05
Hybrid (1069 steps, combined)	0.5000	0.8950	0.8660	0.7762	00:32 + 0:05

5.4.3 Regularization

Based on our results from the previous section, we compare SMART regularization with the default dropout regularization we've been using. We used the configurations for the top 2 models: Full Sequential (6,1,3) and Hybrid (755 steps, combined) training. For all the SMART runs, we turned off the dropout for the task layers, but kept it at 10% for the BERT embeddings layers. For the hybrid tests, we didn't re-run the first full sequential training. From our regularization experiments, we

Table 3: Train/DEV scores for different regularization techniques.

Training and Regularization	**SST-5** Accuracy	**QQP** Accuracy	**STS** Correlation	**Overall** DEV Score
Full Seq. + 0.1 Dropout	0.9967/0.5032	0.9567/0.8793	0.9743/0.8597	0.7708
Full Seq. + SMART $\lambda = 1$	0.7765/0.4950	0.9897/**0.9068**	0.9621/0.8577	0.7769
Full Seq. + SMART $\lambda = 10$	0.8720/0.4905	0.9872/0.8998	0.9601/0.8372	0.7696
Hybrid + 0.1 Dropout	0.7287/0.5032	0.9340/0.8952	0.9895/0.8702	0.7778
Hybrid + SMART $\lambda = 1$	0.7650/0.5130	0.9360/0.8960	0.9940/**0.8710**	**0.7815**
Hybrid + SMART $\lambda = 10$	0.712/**0.5159**	0.9330/0.8946	0.9880/0.8680	0.7814

conclude that using the hybrid training with SMART and $\lambda = 1$ yields the best results. We included the training accuracy in this table to show how the regularization mitigated the overfitting by reducing the distance between train and dev scores.

5.4.4 Additional Datasets

NLI was trained with a learning rate of 2e-5, BERT dropout probability of 0.1 and for 1 epoch. Dev accuracy for MultiNLI was 0.742 and 0.870 for SNLI after 1 epoch. We added a linear layer followed by softmax classifier and used binary cross entropy loss. For the training, we used the hybrid approach with SMART $\lambda = 1$ From our results, we see no improvement in the overall score for any of the datasets, however, there's a task-specific increase for the AMR and SICK datasets.

5.4.5 Ensemble and Final Model

For our final model, we did one round of single task fine-tuning (STFT) for each task over their entire datasets, with SMART regularization and $\lambda = 20$ to combat potential overfitting caused by this last round of training. All ensembles use majority vote for classification tasks (sentiment, paraphrase) and average for regression tasks (STS). In the case where all predicted values were different, the mode tie-breaker was to take the smallest value.

Table 4: DEV scores for different dataset extensions

BERT Base and Extensions	**SST-5** Accuracy	**QQP** Accuracy	**STS** Correlation	**Overall** Score
$BERT_{BASE}$ (No Finetune)	0.144	0.548	-0.009	0.3960
$BERT_{NLI}$ (No Finetune)	0.173	0.366	0.415	0.4151
$BERT_{BASE}$	0.5130	**0.8960**	0.8710	**0.7815**
$BERT_{NLI}$	0.4860	0.8890	0.8580	0.7681
$BERT_{BASE}$ + SST (Yelp)	0.5090	0.8950	0.8640	0.7787
$BERT_{BASE}$ + SST (AMR)	**0.5150**	0.8900	0.8680	0.7795
$BERT_{BASE}$ + STS (SICK)	0.4950	0.8880	**0.8800**	0.7743

Table 5: DEV scores for our final models and ensembles

Model Configuration	**SST-5** Accuracy	**QQP** Accuracy	**STS** Correlation	**Overall** Score
(1) $BERT_{BASE}$ + Full Seq. + SMART + STFT	0.5100	0.9019	0.8790	0.7838
(2) $BERT_{BASE}$ + Hybrid + SMART + STFT	0.5286	0.8956	0.8890	0.7900
(3) $BERT_{NLI}$ + Hybrid + SMART + STFT	0.5032	0.8901	0.8596	0.7743
(4) $BERT_{BASE}$ + Hybrid + AMR + SICK + SMART + STFT	0.5086	0.8965	0.8755	0.7809
(1,2,3) Ensemble	**0.5368**	**0.9035**	**0.8960**	**0.7960**
(1,2,4) Ensemble	0.5204	0.9034	0.8805	0.7880
(1,2,3,4) Ensemble	0.5195	0.9028	0.8832	0.7880

The model with the best results was the combination of the two $BERT_{BASE}$ models plus the $BERT_{NLI}$-based one. This shows how the NLI embeddings helped to complement the original BERT embeddings, more on this in the analysis section. Our final ensemble achieved a test score of 0.515 (SST), 0.904 (QQP) and 0.880 (STS), and an overall score of **0.786**.

6 Analysis

Figure 2 illustrates a significant decrease in the gap between training and DEV accuracy for the paraphrase and sentiment tasks when SMART regularization is applied. By epoch 1, we observe how SMART regularization limits the model's ability to become complex and over-fit to the training data, resulting in a more generalized embedding that performs better on the DEV data especially for the sentiment task. However, SMART regularization slightly increased this gap for the similarity tasks; we believe that the decreased complexity of the model slightly compromised how the model handles the similarity task in favor of improving its robustness for the sentiment classification and paraphrase detection tasks.

When comparing the BERT base version against the NLI-trained, without fine-tuning, we can see how much the NLI fine-tuning improves the STS task, however, paraphrase task's accuracy is reduced, while sentiment accuracy almost remained identical. Furthermore, when training the full model using hybrid + SMART, we notice that using $BERT_{BASE}$ outperforms $BERT_{NLI}$ in every task. This can mean that training BERT with NLI doesn't provide enough semantic significance to the embeddings to improve our particular tasks. To expand on this analysis, we explored the overlap in correct predictions across the target tasks for each model outlined in table 5. Model 3, $BERT_{NLI}$, produced 56 correct predictions for the sentiment classification task that do not overlap with any other models' correct predictions, with the runner up being model 1 (52 unique correct predictions). Similarly, for the similarity task, model 3 also produced the most correct predictions that no other model correctly identified (49). However, for the paraphrase task model 1 significantly outperformed other models by identifying the most predictions (753) that do not overlap with other models' predictions, the runner-up being model 3 with 333 unique predictions. Given this significant gap, we believed that constructing an ensemble model with models 1 and 3 would strongly outperform existing work since

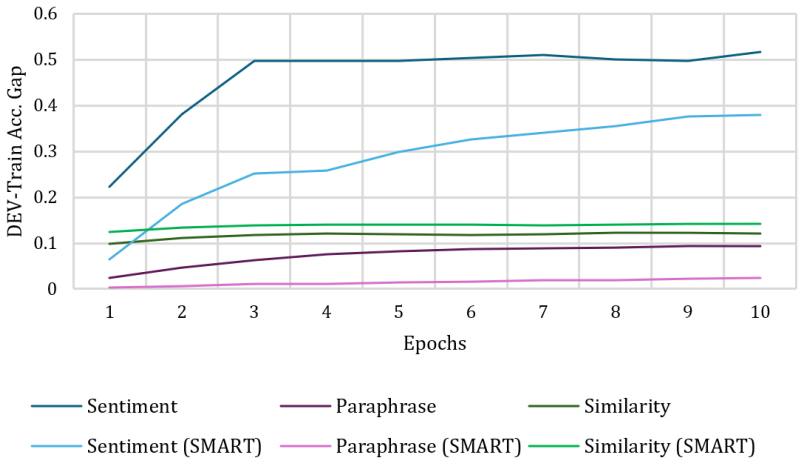

Figure 2: Gap between training and DEV accuracy for hybrid results vs hybrid + SMART ($\lambda = 10$)

the combined understanding of non-overlapping predictions could potentially lead to a more robust ensembled model. The NLI and base model could potentially target different aspects of the sentence to accomplish the task. In addition, the overlap between the correction predictions from model 1 and 3 were not as high compared to that of other combinations of model pairs, which could inform that each model correctly predicted on significantly different aspects of the data. As expected, the (1,2,3) ensemble outperformed other model configurations, likely due to the combined strengths illustrated from this data subset analysis. On the other hand, extending the SST and STS training data with Amazon, Yelp and SICK datasets showed improvement in their respective tasks, however, it also lowered the scores for the other two, resulting in an overall lower score. We didn't expect much increase in SST, given that the datasets we used were labelled based on a different basis when compared to SST. For STS, the SICK dataset also increased the correlation. The SICK dataset criteria for semantic similarity is also different to SemEval, which also affects the lack of performance gains. Also, the sentences from the SICK dataset were much simpler when compared with the ones from SemEval. Overall, we think the SICK dataset can help to fine-tune the last linear layer, but not the BERT embeddings.

7 Conclusion

Our project demonstrated that fine-tuning techniques significantly enhance the performance of the baseline minBERT model on sentence-level tasks. By experimenting with various neural network architectures, optimizers, regularization methods, additional datasets, and training techniques we observed marked improvements. The cross-encoder architecture and hybrid training approach yielded the most substantial gains. Incorporating SMART regularization, additional datasets, and ensembling models further booster performance, with our best model achieving a DEV set score of 0.796 and a Test set score of 0.786. These results underscore the effectiveness of our methods in optimizing the minBERT model and balancing the tradeoff between more data and overfitting.

8 Ethics Statement

Algorithmic bias in BERT models is a ethical consideration that must be taken into account. Monolingual BERT models can contain ethnic bias in different languages based on the historical or social context of the countries in which these languages are used. For example, in the sentence "People who came from [MASK] are pirates", the words with the top mask prediction probability are "Somalia" and "Cuba" Ahn and Oh (2021). To solve this issue, the authors used fine-tuning methods and data augmentation. Another concern in this area, are models that are optimized for generality may struggle with cultural specific language. Specifically with our project, the choice of datasets for fine-tuning could significantly impact how our fine-tuned minBert model handles ethnic bias when it predicts sentiment in a sentence. It is possible that during the fine-tuning process, minBert adapted to consider

certain cultural references or mentions as negative, which can significantly impact text related to this content (such as a review of an international movie). This is why it is important to use additional datasets in order to make the training data more diverse. Another ethical concern stems from the risks associated with wide-scale data collection to build labeled text corpora for BERT training and testing. As we observed with the MultiNLI and SNLI corpora papers, there is a push to construct datasets of novel magnitudes that display promising diversity and training/testing strength to improve models such as BERT . However, in this process, unethical practices may be used to facilitate data collection in unseen scales Victoria Smith (2023). In our project, we extended our use of the STS, SST, and Quora datasets to include MultiNLI, SNLI, and Yelp datasets to finetune with data from a variety of sources. However, as we add on more datasets, we could run into the issue of including data that may have been collected unethically or may contain private information. To mitigate this, we focused on selecting data that has been previously curated and pre-processed by academic researchers to the acceptable standards of the publishing site. In addition to this precaution, the authors recommend pre-processing techniques such as sanitising (removing private information) to protect sensitive information that may have been collected and employed in the training without proper consent from the parties affected. In addition, human evaluators that are employed for labelling tasks should be tasked to identify sentences containing identifying information or sensitive data that will require deidentification.

References

Jaimeen Ahn and Alice Oh. 2021. Mitigating language-dependent ethnic bias in bert.

Archinet AI. 2022. smart-pytorch. `https://github.com/archinetai/smart-pytorch`.

Iain Murray Asa Cooper Stickland. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models.

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

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Carolyn Ashurst Adrian Weller Victoria Smith, Ali Shahin Shamsabadi. 2023. Identifying and mitigating privacy risks stemming from language models: A survey.

Yelp. 2023. Yelp open dataset.