minBERT and Downstream Tasks

Stanford CS224N Default Project

Yvonne Hong
Stanford University
yhong23@stanford.edu

Hodan Farah
Stanford University
hfarah@stanford.edu

Simrin Kalkat
Stanford University
simrin@stanford.edu

Abstract

The focus of our paper is to propose a new set of techniques that can enhance the performance of BERT embeddings on three important and simultaneous tasks: sentiment analysis, paraphrase detection, and semantic textual similarity (STS). Our proposed paradigm is built on the prior work of Sun et al. [1], and it involves the implementation of several extensions, specifically layer-wise learning rate decay, cosine similarity fine-tuning for STS, contrastive learning, additional layers, additional finetuning datasets and modifying weights. Through these techniques, we aimed to address the limitations of existing methods and improve the accuracy and efficiency of BERT embeddings for the aforementioned tasks. We ran 9 experiments on models with individual extensions and combinations of extensions. Our best model outperformed the baseline with accuracies of 0.509 for sentiment analysis, 0.745 for paraphrase detection, and 0.506 for STS. The best model outperformed the baseline on STS by 42%. We learned that in multitask models, added complexity in the model improves accuracies of tasks with larger datasets but causes overfitting for tasks with smaller datasets. We also learned that weighted loss and additional datasets can help improve performance.

1 Key Information to include

Our mentor is David Huang. We have no external collaborators and we are not sharing projects.

2 Introduction

Text classification is an important natural language understanding task and a fundamental milestone in artificial intelligence. BERT, a multi-layer bidirectional transformer, has been trained on plain text for masked word prediction and next sentence prediction tasks. While various techniques for fine-tuning BERT have been proposed, there is still a lack of consensus on which approaches are most effective for different text classification tasks. Sun et. al [1] discusses the methods for optimizing BERT usage in text classification, investigating a variety of fine-tuning and pre-training approaches to elevate its performance in this area. This paper builds on previous literature pertaining to language model pre-training and multi-task learning to explore the effectiveness of BERT fine-tuning for text classification for three downstream tasks: sentiment classification, paraphrase detection and semantic textual similarity.

This paper utilizes a multitask classifier to perform our downstream tasks. Multitask deep learning models are neural network models that can simultaneously perform multiple related tasks using a single network architecture with multiple output layers, each output layer corresponding to a different task [2]. Shared layers of the model learn to extract features that are useful for all the tasks, while the task-specific layers learn to make predictions for each task. Multitask learning has various applications including computer vision, bioinformatics, and more [3]. Although powerful, this technique also comes with challenges which must be overcome [2]. For example, because multitask classifiers perform related tasks, attempting to solve them simultaneously can cause interference between the models, resulting in lower accuracy. Additionally, it is possible that some tasks may have more data

than others, which can lead to bias towards those tasks and poorer performance on the others. The architecture choices of the multitask classifier and general approach must consider these challenges. Overall, our proposed paradigm involves the implementation of several extensions, specifically layer-wise learning rate decay, cosine similarity fine-tuning for semantic textual similarity, contrastive learning, additional layers, additional fine tuning datasets to finetune to our tasks, modifying weights, and adding task specific layers to improve our multitask classifier.

Ultimately, cosine similarity, layer-wise learning rate decay, additional layers for paraphrase detection, additional training on the SICK dataset and finally a weighted loss approach proved to be the most effective at improving the overall accuracy of our model from baseline, as discussed in our results section.

3 Related Work

Several recent studies have explored the effectiveness of pre-trained language models such as BERT for various natural language understanding tasks. Sun et al. are credited for the development of a fine-tuning method for the BERT model for text classification [1]. To fine-tune BERT for text classification, the Sun et al. first added a classification layer on top of the pre-trained BERT model and then fine-tuned the entire model on a labeled dataset for the target task. They also investigated fine-tuning methods for BERT on the target task, including preprocessing layer wise learning decay. Although BERT has achieved impressive results in natural language understanding tasks, prior to this paper there was little research on how to further enhance its performance on target tasks.

Another study done by Reimers and Gurevych [4] proposes a Siamese BERT network, a type of neural network architecture that utilizes BERT embeddings to compare two input sequences and generate a similarity score, for learning high-quality sentence embeddings that can be used for various downstream tasks, including text classification [4]. Our work follows this study's approach of comparing sentence embeddings by applying cosine similarity to improve the accuracy of our semantic textual similarity task. More recent work has explored the effectiveness of contrastive learning for learning high-quality sentence embeddings. For example, Gao et al. proposed SimCSE, a simple contrastive learning method for sentence embeddings that achieved state-of-the-art results on several benchmarks [5]. SimCSE trains sentence embeddings to be close in vector space if they are semantically similar, and far apart if they are dissimilar, leveraging cosine similarity to do so. SimCSE informed how we used contrastive learning to learn more informative sentence embeddings for our multi-task classifier.

Overall, our work builds upon these previous studies by investigating the effectiveness of a framework of techniques that includes cosine similarity, layer-wise learning rate decay, contrastive learning, and additional fine-tuning datasets to improve the accuracy of a multi-task classifier for our three downstream tasks. By incorporating these techniques into our framework, we aim to learn from previous studies and provide a promising analysis of the effectiveness of BERT fine-tuning for text classification and aim to improve the performance of multi-task classifiers for a range of downstream tasks.

4 Approach

4.1 Baseline

Our baseline is a simple implementation of minBERT that retains key BERT architectures such as multi-head self-attention, a transformer layer, and pre-training and uses the Adam optimizer. For more details, we refer the reader to the default project handout. In addition, our multitask classifier was constructed using a dropout layer, and utilized one linear layer per task. Our performance evaluation on the SST, Quora, and SemEval datasets, and using the default parameters serves as our baseline to compare the effectiveness of fine-tuning and other modifications on our model.

4.2 Extensions

4.2.1 Cosine Similarity

Our benchmark accuracy value for STS was relatively low. In an attempt to further fine tune our embeddings to this task, we implemented cosine similarity as an extension based upon Reimers and Iryna Gurevych's paper. Cosine similarity is a widely used metric for measuring the similarity between two vectors, and has been proven effective in comparing sentence embeddings as seen in the paper. Our predict similarity function calculated a cosine similarity score using the pytorch function, and we utilized the mean squared error (MSE) loss function as done in the paper [4].

4.2.2 Layer-wise learning rate decay

Based on Sun et al.'s paper [1], we implemented decreasing decay rate across layers with specific parameters of $\xi = 0.95$ and lr=1.5e-5. In other words, since our BERT model has 12 layers, the last layer would have the learning rate of 1.5e-5 and the first layer has the learning rate of 1.5e-5 * 0.95^{12} = 8.1e-6. We chose the decay factor of 0.95 because Sun's paper [1] discovered 0.95 to be the most effective parameter. We wanted the average learning rate across the layers to be around the default learning rate we were given, which is 1.0e-5. With this extension, our model retains the most information from the most relevant last layers with the higher learning rates, and our model retains less information from the first layers to reduce impact of interference between different tasks.

4.2.3 Contrastive Learning

Based on Gao et al.'s paper [5], we implemented an unsupervised SimCSE, which takes an input sentence and predicts itself with only standard dropout as noise, which serves as minimal data augmentation. We will simply feed the same input into the encoder twice, with creating two embeddings z and z'. We will take the new training objective as:

$$\ell_i = -\log \frac{e^{\text{sim}(\mathbf{h}_i^{z_i}, \mathbf{h}_i^{z_i'})/\tau}}{\sum_{j=1}^{N} e^{\text{sim}(\mathbf{h}_i^{z_i}, \mathbf{h}_j^{z_j'})/\tau}},$$

[5]. The model will then be trained on the new objective. We trained the contrastive learning within each epoch, together with the three task-specific finetuning objectives.

4.2.4 Additional Finetuning Datasets

In our study, we aimed to fine-tune a pre-trained BERT model for the tasks of sentiment analysis and semantic textual similarity (STS) using two small-scale datasets: the Stanford Sentiment Treebank (SST) and the SemEval STS Benchmark. Given the limited data within SST and SemEval, we were concerned about potential overfitting. To address this limitation, we chose to additionally finetune our model by including the SICK_train.txt dataset [6], which contains 10,000 sentence pairs of varying similarity levels on a scale from 0 (unrelated) to 5 (equivalent meaning) which would provide additional data for our semantic textual similarity task. Our goal was to enhance the training data by increasing its diversity and scale, which can help prevent overfitting and improve the model's robustness to generalize to new data.

Moreover, we considered pre-training the BERT model with target-domain data, as suggested by Sun et al.'s [**?**] paper. However, we opted not to pursue this approach, as the pre-trained BERT model we used, 'bert-base-uncased', had already been trained on a vast corpus of English language text, including BookCorpus and English Wikipedia. This pre-training process had already generated robust word embeddings. Therefore, we focused on fine-tuning the pre-trained model using the augmented training data to achieve better performance on our classification tasks.

4.2.5 Modifying Weights

To train our model, we initially used a joint loss function that combined the losses for all three tasks, as follows:

Loss = loss1 + loss2 + loss3

where loss1, loss2, and loss3 are the losses for sentiment analysis, paraphrase detection, and STS tasks, respectively. However, upon evaluating our baseline model, we observed that the loss values for tasks 1 and 2 were significantly smaller than the loss value for task 3. This disparity in loss values inadvertently led to the paraphrase detection task being weighted more heavily than the other two tasks in our overall loss calculation. This imbalance in weighting could lead to a bias towards the paraphrase detection task during training and affect the overall performance of our model.

To account for these different outcome ranges and different dataset sizes for the three tasks, we added weights to the three loss based on the following principle:

Weight(specific task) = SQRT(total number of data/number of data of the specific task) * (1/average magnitude of loss for a batch)

This modification allowed us to balance the importance of all three tasks in our joint training process and obtain more robust embeddings that could generalize better to new data.

4.2.6 Layers

We implemented a linear layer followed by a ReLu activation function, and another linear layer our BERT classifier for each of our three tasks: paraphrase, semantic textual similarity, and sentiment analysis. The linear layer projects the input data onto a higher-dimensional space, allowing the model to capture more complex relationships between the input and output. The ReLu activation function allows the model to introduce non-linearity into the decision-making process. ReLu activation sets negative values in the linear output to zero, while keeping positive values unchanged. This enables the model to capture non-linear patterns in the input data, which may be essential in achieving good performance on the three tasks.

Finally, the second linear layer serves to map the abstract representations learned from the ReLu activation function to the output layer, allowing the model to make predictions based on the complex features learned from the input.

5 Experiments

5.1 Data

We used the Stanford Sentiment Treebank (SST) dataset for sentiment analysis. Stanford Sentiment Treebank (SST) dataset has the split of train (8,544 examples), dev (1,101 examples) and test (2,210 examples). The label choices include negative, somewhat negative, neutral, somewhat positive, and positive. We used the Quora dataset for the paraphrase detection task. The Quora dataset has the split of train (141,506 examples), dev (20,215 examples), and test (40,431 examples). The labels indicate whether particular instances are paraphrases of one another. We used SemEval STS Benchmark dataset for the semantic similarity task. The dataset has the split of train (6,041 examples), dev (864 examples), and test (1,726 examples). The labels indicate similarity between a pair of sentences on a scale from 0 to 5.

In models containing the additional dataset, we also utilized the Sentences Involving Compositional Knowledge (SICK) dataset [6], which contains 10,000 sentence pairs of varying similarity levels on a scale from 0 (unrelated) to 5 (equivalent meaning). We picked this additional dataset to improve the performance of the STS task, and we believed this dataset is the ideal choice because it has the same label structure as our given SemEVAL STS Benchmark dataset.

5.2 Evaluation method

We utilized the evaluations.test-multitask() function on the dev datasets detailed in the default project handout. We used an accuracy metric to evaluate the model's performance on the SST and Quora datasets. When testing the SemEval STS Benchmark dataset, we calculated the Pearson correlation of the true similarity values against the predicted similarity values across the test dataset in order to evaluate our model.

5.3 Experimental details

We utilized default parameters of 10 epochs, batch size of 8, and a hidden dropout rate of 0.3. We used the learning rate of 1e-5 for finetuning, and for models including layer-wise learning rate decay, we used an initial learning rate of 1.5e-5 and a decay factor of 0.95. We trained the above 6 extensions under the method section individually, and we combined some of them to explore a best result model. We evaluated the best model based on the average accuracy of the three tasks.

5.4 Experiments

Model	Description
Model 1	Baseline BERT
Model 2	BERT + Cosine sim (COS) + Layer-wise Learning Rate Decay (LR)
Model 3	BERT + Contrastive Learning (CL)
Model 4	BERT + Additional Dataset
Model 5	BERT + Layers
Model 6	BERT + WeightedLoss
Model 7	BERT + COS + LR + Layers
Model 8	BERT + CL + COS + LR + Layers + WeightedLoss
Model 9	BERT + CL + COS + LR + Layers + WeightedLoss2 (used different weights to Model 8)
Model 10	BERT + COS + LR + Layers (only for Paraphrase task) + WeightedLoss + Additional Dataset

Table 1: Model Descriptions

5.5 Results

Model	Train Accuracy			Dev Accuracy			Dev Avg
	SST	PARA	STS	SST	PARA	STS	
Model 1	0.723	0.955	0.686	0.472	**0.779**	0.357	0.536
Model 2	0.959	0.752	0.814	0.51	0.746	0.431	0.562
Model 3	0.954	0.757	0.879	0.509	0.74	0.378	0.542
Model 4	0.986	0.733	0.764	**0.512**	0.75	0.372	0.545
Model 5	0.956	0.711	0.740	0.507	0.755	0.346	0.536
Model 6	0.894	0.740	0.884	0.504	0.727	0.377	0.536
Model 7	0.766	0.829	0.940	0.511	0.748	0.42	0.560
Model 8	0.788	0.754	0.812	0.5	0.753	0.462	0.572
Model 9	0.921	0.788	0.941	0.5	0.764	0.406	0.557
Model 10	0.988	0.787	0.918	0.509	0.745	**0.506**	**0.587**

Table 2: Model Results

Test Set Result of our best model - Model 10. Average: 0.583 SST: 0.538 Para: 0.747 STS: 0.465

We found that Contrastive Learning decreased our models' performance from baseline. Additionally, we surprisingly found that adding additional task specific layers and a weighted loss approach had no effect on our baseline performance as individual extensions to our minBERT baseline implementation, however these extensions in combination with cosine similarity and training on an additional data set with additional layers improved performance [Table 1][Table 2]. We believe that is because when used individually, the additional layers and weighted losses added to complexity of the model without balancing out the interference between different tasks.

Finally, our best performance was achieved with cosine similarity, layer-wise learning rate decay, additional layers for paraphrase detection, additional training on the SICK dataset and finally a weighted loss approach [Figure 3 in Appendix]. It outperformed the baseline with accuracies of 0.509 for sentiment analysis, 0.745 for paraphrase detection, and 0.506 for STS. It also outperformed the baseline on STS by 42

From the plot of our training metrics for our best model [Figure 3 in Appendix], we also noticed that across the epoch, there is a steady improvement in paraphrase detection accuracy, while the SST

and STS accuracies fluctuated. We believe this observation points to how the model trades off the accuracies between tasks.

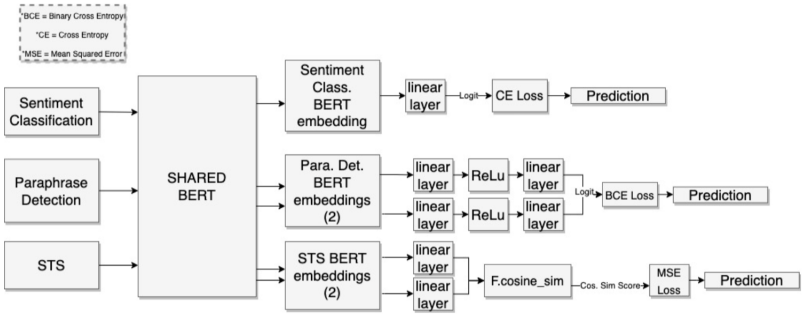

Figure 1: Architecture of the best model - Model 10

6 Analysis

6.1 Comparisons across three different tasks

The accuracy of paraphrase detection is the highest amongst the three tasks for all the models. Overall, we observed the most drastic improvement for STS from the baseline, since in model 10, we can observe a 42% improvement from the baseline STS correlation score. No model in the experiment was able to outperform the baseline on paraphrase detection accuracy, which we believe was because all our extensions to some extent focus on improving the accuracy for SST and STS tasks since they performed very poorly in the baseline. In attempts to improve the SST and STS tasks, we reduced the model's focus on the paraphrase, leading to slightly worse results in paraphrase detection.

SST and STS downstream models are frequently overfitted. 6 out of 9 models we implemented yielded train accuracies of above 0.9 for SST, while the highest dev accuracy on SST in any model is only 0.512. We tried resolving the issue by giving SST a smaller weight when calculating weighted loss, as evidenced by Model 9. Given that our weight calculation is weighting relative to other tasks, we can infer that our model currently prioritizes the SST task and thus trades off reducing overfitting in other tasks for overfitting in the SST task. Hence, we thought that giving SST a smaller weight would prevent overfitting. However, this was still not an effective strategy. The failure of the weighted loss strategy made us believe that the nature to SST's overfitting problem is the small training dataset.

6.2 How Different Strategies Worked on Different Tasks

6.2.1 Adding Complexity Through Layers and Multiple Extensions

We noticed that the model with more layers (i.e. model 5), as well as models with more complexity since they have multiple extensions (i.e. model 8 and 9), tend to do very well in paraphrase detection. These three models have an average of 0.757 accuracy for paraphrase detection, while the rest of the models have an average accuracy of 0.743. However, model 5, 8, and 9 tend to underperform in SST and STS tasks. For example, model 5 is the only model with the accuracy for STS lower than the baseline. These outcomes show us that complexity of the model and complexity of the training objectives helps with paraphrase detection, because of its large dataset size. The complexity helps the model help capture the nuances and complicated information within the large dataset size. On the other hand, since the training set sizes of SST and STS are respectively 0.06 and 0.04 times the size of paraphrase detection training set, the complexity of the model led to overfitting of the SST and STS tasks. For example, for model 9, the train accuracy of SST is 0.921 and the train accuracy of STS is 0.941, which are both much higher than the dev accuracy of the respective tasks. Therefore, we believe complexity of the model helps with accuracy of the task with a big train dataset, while it leads to overfitting of the tasks with small train datasets.

6.2.2 Contrastive Learning

We attempted to improve the BERT embeddings through contrastive learning, but we learned that contrastive learning failed to create any noticeable improvements in any of the tasks (see results for model 3). We believe that is because the pre-trained BERT model we used, 'bert-base-uncased,' had already been trained on a vast corpus of English language text, including BookCorpus (consisting of 11038 unpublished books) and English Wikipedia. Hence, the pre-trained BERT embeddings are already very robust since they are trained on an incredibly large dataset of 3.3 billion words, and further pretraining as a strategy in general would not provide any substantial improvement.

6.2.3 Additional Dataset

The additional SICK dataset used in Model 4 on the STS task minimally increased accuracy. However, when combined with other extensions such as weighted loss, the added SICK dataset substantially improved STS accuracy (Model 10). Our paraphrase task with 100,000+ sentences had a drastically outsized impact on model training against our STS and sentiment analysis tasks with 10,000 sentences. Weighted loss enabled our STS and sentiment analysis tasks with less data to greater influence model learning, reaching parity with the paraphrase task.

6.2.4 Reducing Interference between Tasks

We realized that the biggest challenge in creating a multitask model is interference between different tasks. In an attempt to mitigate this problem, we implemented task-specific layers for our paraphrase task. We considered implementing layers across all three tasks, however as reference previously, adding complexity to the smaller datasets resulted in overfitting, so we chose to focus on exclusively adding layers to our larger Quora dataset. This is evidenced by our results from Model 5 (BERT + Layers) which shared a similar accuracy rate to our baseline as compared to Model 10, which leverage task-specific layers, and produced a significantly higher average dev accuracy rate. This strategy helped our multitask BERT classifier minimize interference between tasks by allowing our model to focus on learning task-specific features and representations without being influenced by irrelevant information from the other tasks.

Another strategy we used was redistributing the weight factor for losses of each task to address the data imbalance and different loss values between tasks. We realized that the paraphrase task was interfering with the performance of other models. We had expected STS and SST task accuracy increases as we added additional components to BERT, but weren't seeing improvements. We realized that this was likely due to the paraphrase task having more influence by being responsible for most of the loss value, as we were training the model initially by adding the losses across the 3 tasks. After realizing this was suboptimal, implementing weighted loss enabled STS performance to improve and reach its best performance at Model 10, a significant improvement compared to any prior model's performance on STS.

6.3 SST Dev Results Analysis

We decided to further analyze the prediction results of the SST sentiment analysis task to further understand the behavior of our best model (model 10). We mapped out the below normalized confusion matrix with the x-axis being the predicted labels and the y-axis being the true labels. We normalized our confusion matrix by dividing the count of each cell by the total number of instances in the corresponding actual class. From the confusion matrix, we noticed that the model gets the prediction wrong the most frequently when the ground truth label is 2. That is likely because the model only has a single linear layer for this downstream task, which can not capture the intricacies of the task of determining whether a BERT sentence embedding is neutral or somewhat positive. The BERT model is more equipped to detect strong emotions through strong signal phrases, therefore making detecting the sentiments on the end of the spectrum easier.

We also noticed that for sentences with ground truth labels of 0 or 4, there is a moderate chance that the predicted label is one off, but the probability of the prediction being more than 2 away from the true label is incredibly low. We believe that is because the model has learned to detect a general sense of positive vs. negative in the training, but it has not learned to distinguish between somewhat positive and positive. This observation makes us think that our current model might have a very high accuracy rate for binary sentiment analysis.

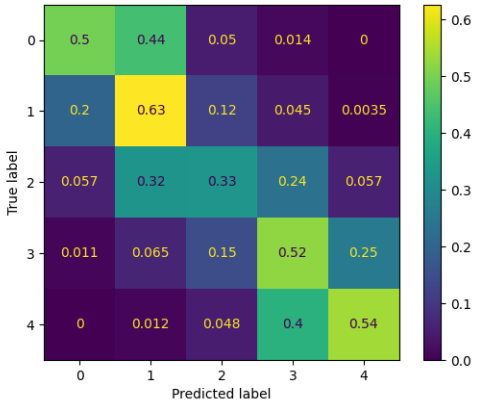

Figure 2: Confusion Matrix of STS Task Accuracies

7 Conclusion

In this project, we aimed to optimize the implementation of a multitask BERT classifier on paraphrase detection, STS and sentiment analysis tasks by leveraging six extensions: layer-wise learning rate decay, cosine similarity fine-tuning for STS, contrastive learning, additional layers, additional finetuning datasets and modifying weights. Our best model outperformed the baseline with accuracies of 0.509 for sentiment analysis, 0.745 for paraphrase detection, and 0.506 for STS. The best model outperformed the baseline on STS by 42%. From our experiments, we learned that complexity helps with larger datasets (the Quora dataset) but causes overfitting for smaller datasets. Therefore, the best model includes complex non-linear layers for paraphrase detection only and very simple linear layers for STS and sentiment analysis. Task interference could be due to a variety of reasons, such as data imbalance as we attempted to address in our weighted loss approach as discussed, or that generally our architecture was too simplistic to robustly leverage a multitask approach on such similar tasks, namely paraphrase detection and STS.

Limitations of our research include the size of the the additional dataset we used. While it aided in increasing attention toward the SST dataset, it's scale was not comparable to Quora. Additionally, we could have included more data for the STS task to further minimize cross task interference. Moreover, our learning rate parameter was drawn from Sun et al [1]'s work, however for a more robust analysis, we could have performed a grid search to determine the optimal learning rate.

For future work, we'd like to explore model structures that allow more independence between tasks, such as independent gradients. We'd also like to experiment with different complicated structures for paraphrase detection and experiment with different weights for weighted loss to learn the tasks with smaller datasets fully, and consider a more complex architecture overall, such as a transformer-based architecture, which may be better suited to more complex and similar tasks. Additionally, we'd like to better understand task interference and investigate transfer learning applications by training models for one task, and then evaluating their performance towards other tasks as model performance for the intended task improves.

References

[1] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification?, 2019.

[2] Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017.

[3] Yu Zhang and Qiang Yang. A survey on multi-task learning, 2021.

[4] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[5] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings, 2021.

[6] https://alt.qcri.org/semeval2014/task1/.

A Appendix

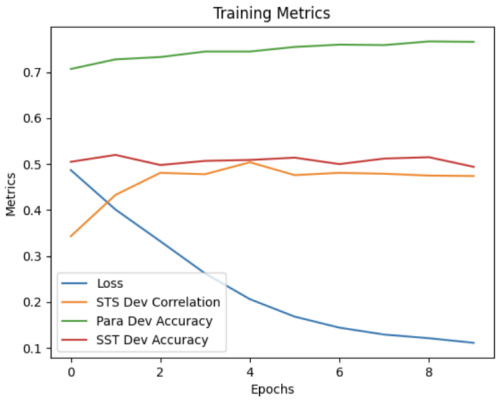

Figure 3: Graph of accuracy + loss over each epoch for model 10