Triple Threat: Exploring Multi-task Training Strategies for BERT

Stanford CS224N Default Project
Mentor: Arvind Mahankali | **External Collaborators**: NA | **Sharing project**: NA

Sally Zhu
Department of Computer Science
Stanford University
salzhu@stanford.edu

Akaash Kolluri
Department of Computer Science
Stanford University
akaash@stanford.edu

Abstract

Multi-task learning models are susceptible to catastrophic forgetting and are unable to retain knowledge of older tasks as they are trained on new tasks; preserving knowledge creates richer, more versatile embeddings. We re-implement minBERT, then explore methods of multi-task fine-tuning to create embeddings that can simultaneously succeed on three downstream tasks: sentiment classification, paraphrase detection, and semantic textual similarity. We explore training orders, attention mechanisms, loss function aggregations, layer-based training, and hyper-parameter settings in order to maximize accuracy across all three tests. We find batch-based round-robin training combats catastrophic forgetting and preforms comparably to indiviudal fine-tuned models. Further, we find that loss-function weighting helps balance knowledge between tasks during training. In our layer-wise training experiments, different layers of BERT yield maximal accuracy for each task, corroborating previous works suggesting that different layers of BERT encode different linguistic features. For tasks that require comparing two sentences, we find that jointly embedding them performs better than separately embedding. With this knowledge, we ultimately train a model comprised of a single BERT body and three small classification heads that achieves a test accuracy of 0.783; through our training mechanism, our BERT model can create embeddings that are more versatile and robust across tasks.

1 Introduction

We explore the problem of multi-task learning, in which one model is trained and optimized to perform well for multiple tasks. With an implementation of the transformer-based minBERT model, we explore different methods of fine tuning on three tasks: sentiment analysis, paraphrase detection, and semantic textual similarity [1]. These three tasks are all applicable in the real world, i.e. one could use sentiment analysis to sort top movies based on film reviews on IMDb or STS to group news articles in a feed based on headlines.

We investigate whether multitask learning on training data from three tasks can create richer embeddings for minBERT and improve performance all around, which Sun et al. (2019) have found to hold on some text classification tasks [2]. We explore various techniques to combat catastrophic forgetting (ie. when a fine-tuned model erases pre-trained knowledge (Sun et al. 2019 [2])) and to simultaneously optimize performance on all three tasks.

2 Related Work

We implement a BERT transformer model described in Devlin et al. (2018), which encodes rich sentence embeddings [3]. Sun et al. (2019) investigate methods for fine tuning BERT for text classification tasks including sentiment analysis [2]. They describe techniques for combating catastrophic forgetting involving decaying learning rates and layer-based training in single-task settings, which we explore and improve upon in a multi-task setting.

We are also inspired by Howard and Ruder (2018), and Jawahar et al. (2019), who suggest that different layers in a BERT model capture different semantic and syntactic information, which may be more or less relevant for different tasks [4, 5]. We do exploration related to this and suggestions from Sun et al. (2019), with original ideas for training for multiple tasks [2].

Finally, we are also interested in loss function weighting across the tasks, in which Bi et al. (2022) and Kendall et al. (2017) find that proper scaling of loss functions by a weighted sum improves performance across tasks [6, 7]. We conduct our own related experiments to improve loss function weighting.

3 Approach

3.1 Model (minBERT) and Optimizer (Adam)

We train a minBERT model using an Adam Optimizer (a gradient-based optimizer that uses adaptive learning rates) to perform three tasks: sentiment analysis, paraphrase detection, and semantic textual similarity, according to the default project handout [1]. Our model has twelve transformer layers with linear classification layers for each of the three tasks. We also apply a dropout layer with a dropout rate of 0.3 after the BERT layers. Unless otherwise noted, we always use a learning rate of $1e-5$.

We use a different classification head and loss function for each of the three tasks. For sentiment analysis, trained on the Stanford Sentiment Treebank, we feed the [CLS] token embeddings into a linear layer with five output classes, each representing a rating of 1 through 5 (negative to positive) [1]. We use cross entropy loss on the logits for back propagation. For paraphrase detection, we experimented with feeding embeddings of both sentences into a linear layer (with double the hidden units), or by feeding a concatenation of the sentences into BERT first, described in Section 4.2. Both would output one logit, and as this is a binary classification task, we use binary cross entropy as our loss function. For STS, we also experiment with the embedding schemes, and the final linear layer outputs one value, and we use mean squared error loss.

3.2 Data and Evaluation

We use the provided train and dev datasets from the default project for each task [1]. We use the provided multitask evaluation code, which computes percent accuracy for SST and Paraphrase, and pearson correlation coefficient for STS. In particular, for SST, we apply a sigmoid function to the logit returned and round to 0 or 1. For paraphrase detection, we take the maximum argument out of the five logits returned, to get the predictions.

To evaluate dev set performance and weight tasks equally, we compute the dev score as $\frac{1}{3}\left(\text{SST accuracy} + \text{Paraphrase accuracy} + \frac{1}{2}\left(1 + \text{ STS corr. coeff.}\right)\right)$

3.3 Baselines

Our baselines are BERT models fine-tuned on each task individually, with performance reported in Table 1. We use the single classification layers and loss functions described in Section 3.1 and a batch size of 16 and train for 10 epochs. For sentence-pair tasks (Paraphrase and STS), we use our basic attention mechanism (see Section 4.2) of embedding both sentences with BERT, then concatenating them to pass into the linear layer.

4 Experiments

4.1 Training Order

We conduct experiments related to the training order of the three tasks. We first test sequential training: we fine-tune the full model on 10 epochs of the SST dataset, followed by 10 epochs on the Quora dataset, and then 10 epochs on the STS dataset. We suspected this would provide average results, possibly with catastrophic forgetting (ie. "forgetting" earlier training data), especially for the SST task.

To combat catastrophic forgetting, we tested two more training methods. First, we tried per-epoch round robin training: alternate between training tasks between every epoch (1 epoch SST, 1 epoch Paraphrase, 1 epoch STS, repeated for 10 iterations). Second, we tried per-batch round robin training: alternate training tasks between each batch (still training for 10 epochs total, within each epoch we repeatedly train 1 batch of SST, 1 batch of Paraphrase, then 1 batch of STS). We believe these approaches will reduce catastrophic forgetting, especially for the SST task, as the model is trained on task-relevant data through the last epoch. In all experiments, we use a batch size of 16 for each task.

4.2 Attention & Embedding Scheme

Next, we conduct experiments related to the attention scheme for the two tasks involving two sentence inputs, specifically whether to concatenate the sentences via input ids and apply BERT or whether to concatenate the two individual BERT embeddings.

Our first mechanism is to first compute the BERT embeddings of sentence 1 and sentence 2 and concatenate the two embeddings to a length $2 *$ hidden size. Then, we apply a linear classification layer. However, this decreases BERT's ability to recognize interactions between the two sentences, which are likely neccesary for paraphrase and similarity tasks. So, we tried a second mechanism, which is to first concatenate the input ids of the two sentences (and with truncating or padding if necessary), feed into BERT, and apply a linear classification layer on the joint embedding. We tested pure concatenation and also concatenation with "[SEP]" in-between (which was used in BERTs original training).

4.3 Weighting Loss Functions

As described in section 3.1, we use three different loss functions for the three tasks, as the datasets and test sets are all different. For our batch round robin models, we perform one backpropagation step with the Adam optimizer after one batch of each of the three tasks, rather than a backpropagation step after each batch per task. This is because the Adam optimizer has a gradient decay factor per call that would decay faster than expected with additional calls [1]. Given the three losses, following Bi et al. (2018), we optimize our joint loss function, which is the sum of each of the three losses [6].

By using different loss functions, however, then the relative magnitude of each task loss may be largely different. For example, after training with unweighted losses for each task, after 10 epochs, we find the losses are:

sst loss :: 0.189, para loss :: 0.019, sts loss : 0.090.

We can see the paraphrase loss in particular is much smaller than the other two losses.

Previous studies including Kendall et al. (2017) find that performance of multitask models is highly dependent on the weighting of such losses [7]. Following Kendall et al. (2017), we use a linear weighted sum of the losses for our forward pass, with weights $w_{SST}, w_{paraphrase}, w_{STS}$:

$$L = w_{SST} \times L_{SST} + w_{paraphrase} \times L_{paraphrase} + w_{STS} \times L_{STS}.$$

Given this, we manually rescale and test different factors for SST CE loss, BCE loss, and STS MSE loss, using the second attention scheme (concatenate inputs, then embed), reporting our results in section 5.3.

4.4 Decaying Learning Rate

Previous work has shown that BERT layers encode differing, meaningful linguistic properties of text (eg. surface, semantic, syntactic) [2, 4]. Motivated by this, we explore various methods of layer-based fine tuning in order to improve the performance of our multitask model and further prevent catastrophic forgetting between tasks.

First, we implement the layer-wise learning decay proposed Sun et al. (2019), where they use a decay factor $\xi < 1$, and the learning rate η^l of the l-th layer of the BERT model is $\eta^{k-1} = \xi \cdot \eta^k$, which decreases for $\xi < 1$. Then, the parameters are updated with: $\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta)$, the usual update formula [2] with different learning rates η^l for different layers. Hence, the weights in the top layers of BERT are updated with a larger learning rate than weights in the lower layers of BERT. Sun et al. (2019) finds that such decaying learning rates, specifically with $\eta = 0.95$ improves performance for a single task classifier (IMDb dataset) [2].

4.5 Layer-based Training

Sun et al. (2019) also investigate model performance across individual layers of BERT. In particular, they freeze all 11 layers of BERT except for one and finetune the single layer, then evaluate performance, to determine which layers encode the most impactful information [2].

While they explore this for single tasks, we extend this idea to our multitask model. We finetune individual BERT layers for each of the three tasks independently (along with the final classification layer) and investigate which ones lead to highest performance. From these results, we run experiments by increasing learning rates for the specific layers for each task, to attempt to train specific layers to better encode information for specific tasks. We select the top four layers that were most accurate for each task, choosing four because of the split of three tasks out of 12 layers. Sun et al. (2019) also find best performance in their experiments as well by selecting four layers to train [2].

We perform a selective learning rate sweep, for different values of α and β, where α is the scaling factor for the eight lower-performing layers, and β is the scaling factor for the four higher-performing layers: $lr_i = \alpha \cdot 1e - 5$ if i is one of the lowest-performing 8 layers, and $lr_i = \beta \cdot 1e - 5$ if i is one of the highest-performing 4 layers.

4.6 Additional Linear Classification Layer

We also test using an additional linear classification layer for each task and sweep over the number of hidden units between the two linear layers, to explore the potential of a more robust final classification layer architecture. We use a RELU activation. Our goal is to create a more robust read of the embedding space that a linear layer alone would not catch.

5 Results

5.1 Training Order Results

Table 1 shows model performance with the different training orders we tested. The plots in Figure 4 of Appendix A show training metrics for the three tasks for the three different multitask training orders over training epochs. We can see in Figure 4(a), which was sequential training, that accuracy for sentiment classification decreases over training epochs for the other tasks, which is evidence of catastrophic forgetting. In Figure 4(b), epoch round robin, there is fluctuation of performance, especially paraphrase detection, between epochs. In Figure 4(c), batch round robin, the performance of all three tasks improve smoothly and concurrently.

Model	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
Single Task Models (Baseline)	0.511	**0.763**	0.345	—
Sequential	0.411	0.691	0.361	0.594
Epoch Round Robin	0.489	0.683	0.354	0.616
Batch Round Robin	**0.520**	0.740	**0.375**	**0.649**

Table 1: Model performance with different multitask training orders.

We achieved the highest dev score with the batch round robin model, as expected, with a score of 0.649 at 8 epochs. Scores for all tasks of the three models and single-task baselines are reported in Table 1. In the batch round robin model, each task also has high performance (outperforming the baseline for SST and STS and performing comparable to the paraphrase detection baseline), suggesting our strategy mitigates catastrophic forgetting.

5.2 Attention & Embedding Scheme Results

We report results from the attention and embedding schemes in Table 2. We find that concatenating the inputs (without a [SEP] token) improves performance for the sentence-pair tasks (paraphrase detection and STS), with accuracy increases from 0.740 to 0.782 and from 0.375 to 0.846, respectively. This is likely due to the model learning interactions between similar words in the sentence pairs (see Figure 5 of the Appendix B for a diagram of how our mechanism encodes richer interactions). Although there is a decrease in SST performance, from 0.520 to 0.506, the dev score increases from 0.649 and 0.737. We use this attention scheme for **all future experiments.**

Scheme	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
concatenate embeddings	**0.520**	0.740	0.375	0.649
concatenate inputs (then embed)	0.506	**0.782**	**0.846**	**0.737**
concat. inputs with [SEP] (then embed)	0.491	0.776	0.821	0.726

Table 2: Model performance with different embedding schemes.

We find that using a separator token also greatly improves performance for the STS task only yet slightly trails in overall dev performance, so we continue with the second scheme from the table.

5.3 Weighted Loss Functions Results

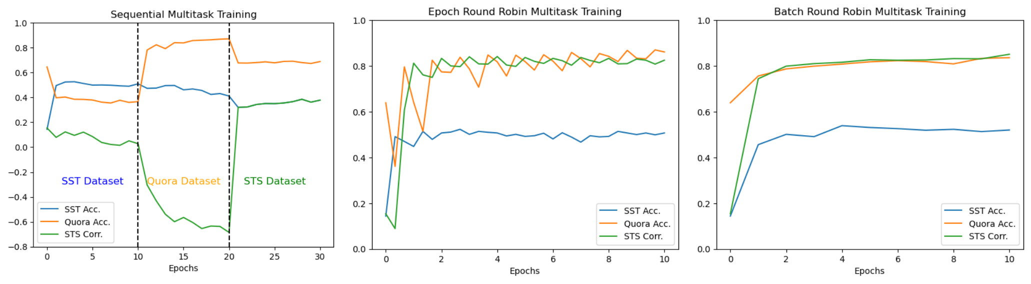
Our manual sweep of $w_{SST}, w_{paraphrase}, w_{STS}$ is reported in Table 9 of Appendix C. We find that a weighting of $w_{SST} = 0.1, w_{paraphrase} = 2, w_{STS} = 0.4$ achieves highest dev score after 10 epochs. Unless otherwise noted, we carry this loss function weighting through future experiments.

Loss scheme	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
Unweighted loss	0.506	0.782	0.846	0.737
Weighted sum loss	0.529	0.841	0.850	**0.765**

Table 3: Model performance improvement with weighted sum of loss.

We see that dev score has improved, as well as performance for each of the other tasks. And by examining task-specific performance, we see that the paraphrase detection task performance improved greatly, from 0.782 to 0.841. We suspect this is because the paraphrase task loss values were at much smaller scales compared to the other tasks and when summing losses directly, $L_{SST} + L_{paraphrase} + L_{STS}$, would be underweighted in the gradient update. Hence weighting the tasks equally in backpropagation and the gradient updates would improve performance.

We also re-run the training order experiments with the updated attention scheme and weighted loss function, with performance for each task across epochs and each training order shown in Figure 1. We see similar evidence of catastrophic forgetting in both Figures 1(a) and 1(b), which further validates our choice of batch round robin training.



(a) Sequential multitask training performance vs epochs.

(b) Epoch round robin training performance vs epochs.

(c) Batch round robin training performance vs epochs.

Figure 1: Training performance for varied multitask training orders.

5.4 Decaying Learning Rate Results

We run experiments for various decay rates η, from 1.0, 0.95, ... 0.75. We implement this by explicitly setting learning rates per layer in our Adam Optimizer. Our goal was to see if varying learning rates across layers, such that lower layers have smaller learning rates, preserves semantic information across BERT layers to create better embeddings.

We report performance for all the decay rates in Table 9 of Appendix C, and of decay values $\eta = 1.0, 0.95, 0.90.$ in Table 4 below (we ran these experiments using loss weights of $w_{SST} = 0.1, w_{paraphrase} = 1, w_{STS} = 0.2$). We find that the highest dev score is achieved with no decaying, and both paraphrase and STS tasks have the highest performance with no layer-based decay as well. Although Sun et al. find better accuracy with a decay rate of $\eta = 0.95$, we suspect this is likely due to single-task versus multi-task model differences [2]. Sun et al. finetune and evaluate on a single sentiment analysis classification task of the IMDb dataset [2].

However, for multitask models, it is possible that different transformer layers encode different information that are more or less important for each diverse task. For example, the STS task increased in performance with decay rate $\eta = 0.95$, which aligns with Sun et al.'s findings on their same task [2]. However, this is not true of the other two tasks, so applying a learning rate decay equally did not help dev performance overall.

Decay Rate	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
1.00	0.520	**0.836**	**0.851**	**0.761**
0.95	**0.528**	0.830	0.846	0.760
0.90	0.505	0.819	0.835	0.747

Table 4: Model performance using different layer-based decay rates.

5.5 Layer-based Training Results

We report the results of our layer-based training experiments in Table 5 (training single BERT layers). The experiments were run with the architecture described up to Section 5.2, with no decaying learning rate, and for 10 epochs and with a learning rate of 1e-5.

BERT Layer trained	SST Acc.	Quora Acc.	STS Corr.
Layer 0	0.416	0.722	0.808
Layer 1	0.418	0.781	0.815
Layer 2	0.457	**0.794**	0.808
Layer 3	0.439	**0.796**	0.808
Layer 4	0.453	0.782	<u>**0.837**</u>
Layer 5	0.437	0.772	**0.827**
Layer 6	0.471	0.751	**0.825**
Layer 7	0.456	**0.792**	**0.828**
Layer 8	**0.497**	<u>**0.803**</u>	0.823
Layer 9	**0.502**	0.770	0.808
Layer 10	<u>**0.503**</u>	0.783	0.788
Layer 11	**0.490**	0.784	0.778

Table 5: Model performance when only training a single BERT layer, independently for each task.

We bold the four layers with the top performance per task. We find that the final BERT layers are optimal for sentiment analysis, and varying middle layers are optimal for paraphrase detection and STS. Hence, we conclude, as Howard et al. (2018) and Sun et al. (2019) and others suggest, that different layers of BERT encode different relevant semantic information [4, 2]. This is further supported by the fact that the paraphrase and STS tasks are related (evaluating similarity of sentences), so they share overlapping middle layers.

Then, we sweep over different values of α and β described in Section 4.5, which scale the learning rates of $1e - 5$ depending on the layer. The learning rate for all classification layers is 1e-5, and we train for 10 epochs. We report our results in Table 11 of Appendix D. We find that the highest performance was with $\alpha = 1$ and $\beta = 2$, with a dev score of 0.763, which was a slight improvement over our dev score of 0.761 without changing learning rates. In later iterations of long-training, however, we find that this trend doesn't hold.

Hence, we conclude that layer-based training may not work for our objective. We suspect that future work could more robustly calculate weightings by layer may speed up convergence for multitask models and improve accuracy.

5.6 Additional Final Classification Layer

We test adding an additional linear classification layer on our (final) model trained with batch round robin, the best attention scheme, weighted loss function, and no differences in learning rate between layers. Given that an additional linear classification layer would increase converge time, we test this both for short training (10 epochs) and long training (50 epochs).

Our hidden unit sweep for additional classification layer is reported in Table 13 and Table 14 of Appendix F, and we find that 120 hidden units is the best performance, with a dev score of 0.777 in the 50 epoch case.

5.7 Final Model Results

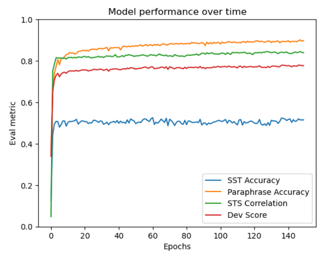
For our final model sweep, we do not conduct an exhaustive aggregation of all of the possible tunings, but combinations that performed well in previous experiments. We ran these tests for 200 epochs, because we noticed experimentally that dev score increased with more epochs (paraphrase accuracy in particular). A full list of tested configurations is available in Appendix G. Over the course of long training, we find that using batch round robin training, concatenated input embeddings, an extra 120 hidden unit linear layer, a learning rate of 1e-5 across all layers, and (0.1,1,0.2) loss weighting trained for 150 epochs achieved the best results. This corresponds to Config A described in Table 15 and 16 of Appendix G. Table 3 summarizes this final model's performance on the train and test set, compared to our intial baselines.

Metric	SST Accuracy	Quora Accuracy	STS Correlation	Overall Score
Development Set	0.531	0.897	0.839	0.783
Test Set	0.534	0.889	0.851	0.783
Baseline	0.511	0.763	0.345	——

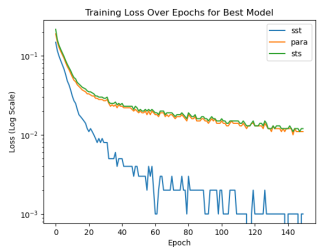
Table 6: Best Model Performance Metrics.

6 Analysis

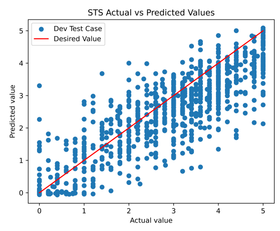
We analyze several metrics of the final model. First, we plot model performance and loss across epochs, shown in Figure 2(a) and (b). We see that dev score is still increasing slightly and loss is still decreasing through 150 epochs (where it achieves the maximum and minimum respectively).



(a) Final model performance over epochs.



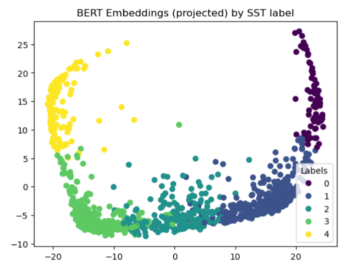
(b) Final model loss over epochs.



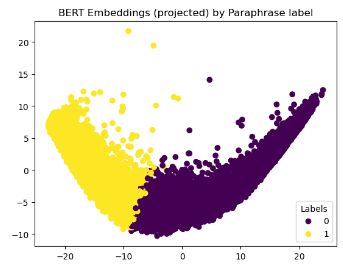
(c) STS Actual vs Predicted labels.

Figure 2: Model performance analysis.

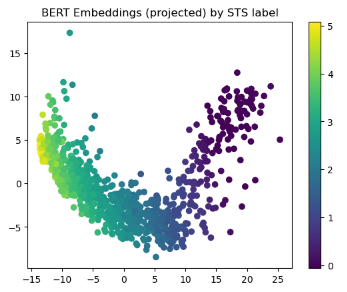
Next, we investigate our BERT embeddings related to the tasks. We performed principal component analysis of the embeddings our BERT layers produced on the dev set data, and we color the data points based on the labels. The plots are shown in Figure 3. We see that there are clear divisions between the labels (discrete or not) for each of the three tasks, which demonstrates that much of the task classification is contributed through the BERT embeddings, and not just the final linear layer. Thus, our BERT embeddings are robust across all three downstream tasks.



(a) BERT Embeddings (projected) by SST label.



(b) BERT Embeddings (projected) by Paraphrase label.



(c) BERT Embeddings (projected) by STS label.

Figure 3: PCA of BERT embeddings for all three tasks.

Then we conduct error analysis via confusion matrices or plots for the tasks, shown in Table 7 and Figure 2(c). We see by inspection that for sentiment analysis, most of the errors are classifications where the prediction is one off from the true value. We see this in the plot for STS as well, that the slope of 1 is true and there are not extreme cases where the prediction differs from the true value. This is reasonable as distinguishing between adjacent sentiment classes or textual similarity values is likely very difficult, and we can see that our model is still performing well even when making errors.

Pred. \ True	0	1	2	3	4
0	45	76	15	3	0
1	26	202	43	17	1
2	2	61	85	74	6
3	1	17	36	170	55
4	0	3	9	70	83

(a) Confusion matrix for SST.

Pred. \ True	0	1
0	23252	2285
1	2122	12770

(b) Confusion matrix for Paraphrase detection.

Table 7: Confusion matrices for SST and Paraphrase task.

Finally, we do qualitative analysis to identify types of errors our model is frequently making. Appendix H contains annotated error examples by case. We notice three large themes of error causes: 1] close word matches causing the model to over predict similarity (eg. "We are meant to explore." and "We are meant to move." are predicted as overly similar) 2] inclusion of terms likely not in the corpus (eg. the term "WAAAAAAAY" confuses the model), and 3] subjective decisions, i.e the classification of sentence comes down to arbitrary interpretation (eg. "how does it feels to be in love?" and "how does it feel to be doing what you love?" are labeled as paraphrases, but could be considered by some as different questions).

7 Conclusion

In this paper, we presented an analysis of various training strategies to improve the accuracy of a BERT multitask classifier, with an emphasis on retaining knowledge between tasks so that our BERT can create more versatile, rich embeddings. Most notably, we find that batch-based round robin training and loss function weighting are effective strategies for simultaneous task training. Ultimately, we achieved an overall test score of 0.783 with our best model. Through our layer-based analysis, we verify previous claims that different layers of BERT encode different linguistic features. A limitation of our work is the limited representatives of our text classification tasks. Although our training strategy performed well on our defined tasks, these strategies may not generalize to multi-task classifiers trained for a different set of tasks. Another limitation of our work's final model is the long epoch training time – our models achieve comparable performance over shorter epoch training time, but we do note that continuous training does *slightly* improve the accuracy. Future work could use our analysis of layer-wise training of BERT to test more robust settings for learning rates that could speed up training time and more effectively balance knowledge across layers for different tasks.

Team Contributions: Sally did the training order, attention mechanism, loss-weighting, and decaying learning rate experiments. Akaash did the layer-based training methods, additional classification layer tests, parameter sweeps, long-model training tests, and error analysis. We collaborated on all tasks.

8 Ethics Statement

Our data come from human text and thus our embeddings likely inadvertently encapsulate social bias (eg. gender and racial bias) that could affect downstream training. It is also well documented that the default BERT weights that we start from contain gendered bias [8]. In the context of our project, this is especially important because training on even a single task with a biased dataset could cause bias to manifest across the shared model layers and therefore all tasks. Further, our evaluation of "robustness" and "versatility" is inherently limited and ignores this bias. One possible solution, based on our research, is to train BERT specifically on debiasing tasks. For example, the task could be binary analogy predictions that punish gendered analogies. Given we show that our methods of training preserve knowledge between tasks, this could be effective in creating robust embeddings that are less biased, even when fine-tuned for later tasks. Bolukbasi et al. propose another method for "de-biasing" gendered embeddings by subtracting a gender embedding direction to normalize non-gendered words [9].

Another ethical concern of our research is data privacy. Our proposed research methods allow for information and knowledge retention across a diverse set of training tasks and data. A principal implication of this is that training BERT on large datasets and tasks is beneficial. This has potential for grave consequences for user data privacy. Institutions may begin further tuning models on larger more expansive datasets, without properly thinking through the ethics of the user data and privacy. Our model, given that it effectively retains this knowledge, could inadvertently retain knowledge of information that poses a privacy risk (eg. sensitive data, copyrighted information) that could manifest in downstream tasks. To mitigate this, first we are committed to ethical sourcing of datasets and confirmation that they are sourced with user consent (and don't contain sensitive personal information). Second, we propose using methods of data preprocessing and augmentation in order to actually blur/remove private and sensitive data as a safe guard against any potential information leaks.

References

[1] CS224N Staff. Cs 224n (spring 2024) default final project: minbert and downstream tasks. 2024.

[2] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification?, 2020.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[4] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification, 2018.

[5] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics.

[6] Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. Mtrec: Multi-task learning over bert for news recommendation. In *Findings*, 2022.

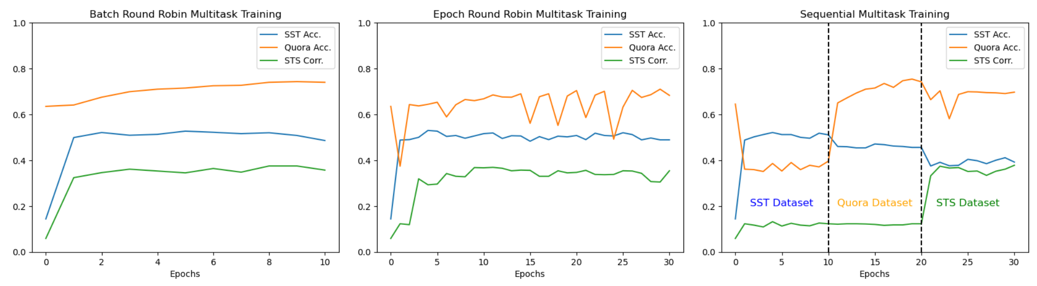
[7] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *CoRR*, abs/1705.07115, 2017.

[8] Rishabh Bhardwaj, Navonil Majumder, and Soujanya Poria. Investigating gender bias in bert. *Cognitive Computation*, 13(4):1008–1018, 2021.

[9] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29, 2016.

A Training Order Results

For our original training order tests (before the updated embedding mechanism and loss weighting), we graph the accuracy for each task over epochs.



(a) Sequential multitask training performance vs epochs.
(b) Epoch round robin training performance vs epochs.
(c) Batch round robin training performance vs epochs.

Figure 4: Training performance for varied multitask training orders, without improved attention mechanism.

We re-conduct our training order analysis after modifying the attention mechanism and loss weighting. The accuracy results after this modification are as follows (a graph of this data is in section 5.3):

Model	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
Single Task (Improved Baseline)	**0.524**	**0.870**	**0.857**	–
Sequential	0.378	0.688	0.856	0.665
Epoch RR	0.507	0.861	0.825	0.760
Batch RR	0.520	0.836	0.851	**0.761**

Table 8: Model performance using different training orders with improved attention mechanism.

B Dual Encoding Attention

Through dual encoding, we encode richer interactions between sentence pairs.

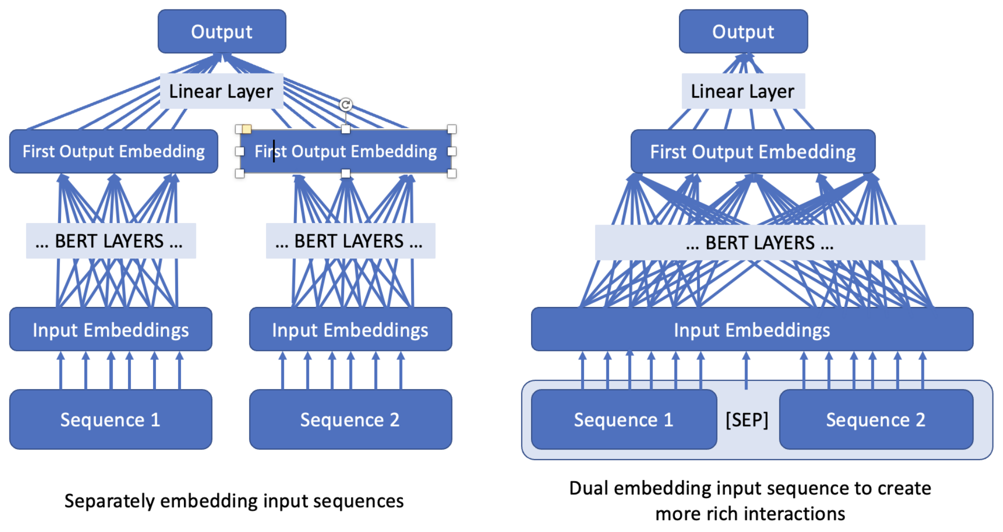


Figure 5: Diagram of encoding mechanisms.

C Loss Function Weighting

We preform a sweep over various loss function weightings and explore their performance over 10 epochs.

Weighting ($w_{SST}, w_{paraphrase}, w_{STS}$)	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
0.1, 1, 0.1	0.524	0.835	0.430	0.760
0.1, 1, 0.2	0.520	0.836	0.851	0.761
0.1, 2, 0.4	**0.529**	0.841	0.850	**0.765**
0.1, 3, 0.6	0.515	**0.842**	0.852	0.761
0.1, 1, 0.4	0.517	0.833	**0.857**	0.760

Table 9: Model performance using different loss function weights

D Layer Based Training

We test decay rates from 1.0 to 0.75 (main text only reports from 1.0 to .9), and find the accuracy is very poor for any decay rate below 0.95. This is likely because smaller decay rates end up training lower layers very minimally (since decay is exponential).

Decay Rate	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
1.0	0.520	**0.836**	**0.851**	**0.761**
0.95	**0.528**	0.830	0.846	0.760
0.9	0.505	0.819	0.835	0.747
0.85	0.513	0.821	0.832	0.750
0.80	0.503	0.816	0.830	0.745
0.75	0.505	0.812	0.824	0.743

Table 10: Performance metrics for different decay rates

As described in 4.5, we also test modifying the learning rates layer wise. For each task the learning rate of the top 4 performing individual layers is set as $\beta * lr$, and the bottom 8 performing are set as $\alpha * lr$ where $lr = 1e - 5$ across all tasks.

α	β	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
1	1	**0.520**	0.836	**0.851**	0.761
0	1	0.400	0.811	0.799	0.703
1	2	0.517	**0.851**	0.841	**0.763**
1	3	0.504	0.843	0.837	0.755
0.5	1	0.502	0.845	0.838	0.755

Table 11: Performance metrics for layer-based learning rate scaling

E Learning Rate Sweep

We also preform a sweep over learning rates. While these results suggest that a higher learning rate may preform better when training for 10 epochs, when we trained for a longer number of epochs, this trend did not hold (higher learning rates resulted in more over fitting). We also performed this sweep after we had finished the bulk of our experiments, so we didn't end up incorporating this knowledge into many parts of our analysis. Thus, we excluded this analysis from the main text, but provide it here as another test we tried.

Learing rate	SST Acc.	Quora Acc.	STS Corr.
1e-6	0.475	0.781	0.793
5e-6	0.509	0.814	0.835
1e-5	0.506	0.836	0.843
2e-5	0.517	0.839	0.848
4e-5	0.501	0.845	0.851
8e-5	0.480	0.839	0.848
16e-5	0.476	0.809	0.834

Table 12: Performance with different learning rates after 10 epochs.

F Hidden Layer Sweep

We add an additional linear classification layer for each of the three tasks, and perform a sweep on the number of units between the two classification layers. There is a ReLU activation function between the two linear layers. We experiment using 10 epochs and 50 epochs, given that the extra layer may increase convergence time.

With 10 epochs, we find the following:

Hidden Size (10 epochs)	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
No Hidden Layer	0.520	0.836	0.851	0.761
10 Hidden Units	0.519	0.836	0.848	0.760
20 Hidden Units	0.509	0.832	0.843	0.754
40 Hidden Units	0.491	0.836	0.851	0.751
80 Hidden Units	0.525	0.832	0.852	0.761
120 Hidden Units	0.518	0.838	0.849	0.760

Table 13: Sweep over hidden layer units at 10 epochs

With 50 epochs, we find:

Hidden Size (50 epochs)	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
No Hidden Layer	0.510	0.869	0.858	0.769
10 Hidden Units	0.503	0.861	0.853	0.764
20 Hidden Units	0.511	0.870	0.844	0.768
40 Hidden Units	0.511	0.871	0.852	0.769
80 Hidden Units	0.517	0.867	0.844	0.769
120 Hidden Units	**0.520**	0.**876**	**0.867**	**0.777**

Table 14: Sweep over hidden layer units at 50 epochs.

G Final Model For Long Epoch Training

For our final model sweep, we do not conduct an exhaustive aggregation of all of the possible tunings, but combinations that performed well in previous experiments. We ran these tests for 200 epochs because we noticed experimentally that the dev score increased with more epochs (paraphrase accuracy in particular). A full list of tested configurations is available in Appendix G. For long training, we find that using batch round-robin training, concatenated input embeddings, an extra 120 hidden unit linear layer, a learning rate of 1e-5 across all layers, and (0.1,1,0.2) loss weighting trained for 150 epochs achieved the best results. This corresponds to Config A described in Tables 15 and 16 of Appendix G. Table 3 summarizes this final model's performance on the train and test set, compared to our initial baselines.

Model	Peak Epoch	SST Acc.	Quora Acc.	STS Corr.	Overall Dev Score
Config A	150/200	**0.531**	0.891	0.853	**0.783**
Config B	149/200	0.516	**0.905**	0.845	0.781
Config C	194/200	0.513	0.892	**0.860**	0.778
Config D	142/150	0.514	0.892	0.852	0.777
Config E	127/200	0.510	0.900	0.856	0.779

Table 15: Peak model performance for various configurations

The following are the details of each model configuration. All models by default use the changes mentioned in the paper. Specifically, they all contained a default learning rate of $1e-5$, batch size of 16, batch round-robin training, the modified embedding mechanism (concatenate inputs and then embed them), and weighted loss (with coefficients described below).

Model	**Details**
Config A	1] Extra Linear Layer with 120 Hidden Units 2] (0.1,1,2) Loss Scaling
Config B	1] Extra Linear Layer with 120 Hidden Units 2] (0.1,2,4) Loss Scaling
Config C	1] Extra Linear Layer with 80 Hidden Units 2] (0.1,2,4) Loss Scaling
Config D	1] No Extra Linear Layer 2] (0.1,2,4) Loss Scaling
Config E	1] Extra Linear Layer with 120 Hidden Units 2] (0.1,1,2) Loss Scaling 3] Layer-wise learning rate modification (described in section 4.5) with $\alpha = 1, \beta = 2$

Table 16: Details of different model configurations

H Qualitative Analysis of Errors

We analyze error examples on each classification task. Table 17 explores the Sentiment Analysis Task, Table 18 explores the Paraphrase Detection task, and Table 19 explores the Semantic Textual Similarity Task. After each example, we have added our best guess as to why the error occurred, in bold and in parenthesis.

Class	Over predicted Sentiment by 1	Under Predicted Sentiment by 1	Prediction Off by 2
0	If there's one thing this world needs less of, it's movies about college that are written and directed by people who couldn't pass an entrance exam. **(Subjective decision)**	–	The Iditarod lasts for days-this just felt like it did. **(Unknown word)**
1	You won't like Roger, but you will quickly recognize him **(Subjective decision)**	Without non-stop techno or the existential overtones of a Kieslowski morality tale, Maelström is just another Winter Sleepers. **(strongly negative words "existential")**	Moretti's compelling anatomy of grief and the difficult process of adapting to loss. **(Many negative sentimnet words, like "grief", "difficult")**
2	The film serves as a valuable time capsule to remind us of the devastating horror suffered by an entire people **(Focused too much on first half of text's sentiment)**	Half Submarine flick, Half Ghost Story, All in one criminally neglected film **(colloquial phrase "criminally neglected" interpreted as negative)**	It's a coming-of-age story we've all seen bits of in other films-but it's rarely been told with such affecting grace and cultural specificity. **(Subjective decision)**
3	It's a much more emotional journey than what Shyamalan has given us in his past two movies,and Gibson, stepping in for Bruce Willis,is the perfect actor to take us on the trip. **(Contains strongly positive words)**	Huston nails both the glad-handing and the choking sense of hollow despair.**("Choking" and "hollow despair" thought of as negative, when they are just descriptions of the movie)**	Generally, Clockstoppers will fulfill your wildest fantasies about being a different kind of time traveler,while happily killing 94 minutes. **("killing" misinterpreted in context)**
4	–	It's refreshing to see a girl-power movie that doesn't feel it has to prove anything **(Subjective)**	We haven't seen such hilarity since Say It Isn't So! **(Movie title "Say It Isn't So!" misinterpreted as a part of phrase)**

Table 17: Examples of Sentiment Analysis Classification Errors

Error Type	Error Example
False Positive (not paraphrased, incorrectly flagged as paraphrased)	1] How long can you leave cooked chicken wings out at room temp? | Can you leave raw chicken out all night? **(Very similar terms; confused raw and cooked)** 2] Why computer vision is hard? | Why computer vision is computationally hard? **(Adjective distinction missed)** 3] Where can I get study material for Cisco 300-101 exam? | Which material is useful for the ccnp route 300-101 exam? (**"Cisco" and "CCNP" not in corpus, so it didn't learn a distinction)** 4] How do deaf-born people think? | How do people born blind and deaf think? **(Slight detail change missed)**
False Negative (paraphrased, incorrectly flagged as not phrase phrases)	1] How do I write an Android application? | How do I create an Android application? **(Confusion of around verb)** 2] How can I know about quantum physics? | How much do we actually know about quantum physics? **(Subjective Decision; could be considered not paraphrased)** 3] Which hotel in surat allows unmarried couples to rent rooms? | What hotel in Surat would be safe for unmarried couples, without the harassment of police, hotel staff, and moral police? **(Additional qualifying details)** 4] How does it feels to be in love? | How does it feel to be doing what you love? **(Subjective Decision; Could be Considered Not Paraphrased)**

Table 18: Examples of Paraphrase Classification Errors

Error Type	Error Examples
Over Predicted Similarity $1 < y_{pred} - y_{acc} < 2$	1] P.G. police seeking driver in crash that killed child | Police seek gunmen in New Orleans Mother's Day parade shooting **(Police, seek, and killing are in both texts)** 2] We are meant to explore. | We are meant to move. **(Model viewed move/explore as synonyms, but they are different in this context**
Largely Over Predicted Similarity $y_{pred} - y_{acc} > 2$	1] Work into it slowly. | It seems to work. **(Close word match, but very different word meanings)** 2] Syrian fighter pilot defects to Jordan | Syrian PM defects to Jordan **(Confused by abbreviation "PM")**
Under Predicted Similarity $1 < y_{acc} - y_{pred} < 2$	1] A group of four people in a dinner having cake. | Four people eating dessert around a table. **(Not able to interpret similarity between diner and table)** 2] A swimmer is doing the backstroke in the swimming pool. | A person doing the back stroke in a swimming pool **(Separation of the word "backstroke" into "back stroke", changing meaning)**
Largely Under Predicted Similarity $y_{acc} - y_{pred} > 2$	1] Israel and Hamas to observe brief Gaza truce | Israel And Hamas 'Accept Temporary Truce' **(Confusion over obsere cs accept)** 2] You guys are making this all WAAAAAY too complicated. | You are making this too complicated.**(WAAAAAY not in training corpus)**

Table 19: Examples of Semantic Textual Similarity Classification Errors