Hybrid BERT: Sharing Layers for Multitask Performance with Fewer Parameters

Stanford CS224N Default Project

Jake North
Stanford University
Department of Mathematics
jwnorth@stanford.edu

Jared Weissberg
Stanford University
Department of Computer Science
jared1@stanford.edu

Abstract

This paper proposes several methods to improve miniBERT's performance on semantic classification, paraphrase detection, and semantic similarity tasks. Our primary contribution is a novel weight-sharing scheme that reduces parameter count while maintaining performance. We find that a triplet of BERT models sharing half of their layers can match the performance of three separate models with no weight sharing. This approach substantially outperforms our best single-BERT models, and we find a comparison of 1-BERT, 3-BERT, and hybrid (weight sharing) models to be insightful for understanding the nature of multitask training of a transformer. To further augment performance, we implement a number of smaller experiments: multitask fine-tuning with gradient surgery, various loss functions and dropout configurations, various classification head architectures, and feature transformations.

1 Key Information

- Mentor: Josh Singh
- External Collaborators: No
- Sharing project: No
- Both project members contributed equally to the code and the writeup

2 Introduction

The Bidirectional Encoder Representations from Transformers (BERT) model, described in Devlin et al. (2018), is a language representation model that pretrains deep bidirectional representations from unlabeled text using both left and right context in all layers. Because BERT can be fine-tuned with one additional context layer, it has resulted in a slew of revolutionary state-of-the-art models for tasks ranging from question answering to named entity recognition.

In this paper, we investigate several methods to improve BERT's performance on three related downstream tasks: semantic classification, paraphrase detection, and semantic textual similarity. Rather than relying on one BERT for each task, we first developed a BERT model (`1-bert`) that performed well across all tasks, trained via a multitask approach with a unified loss function. However, multitask models are prone to issues like conflicting gradients and task interference, even after partial solutions like gradient surgery. For a comparison, we trained a model which contained 3 separate BERT transformers, each tuned on a specific task (`3-bert`). This resulted in markedly better performance, likely by alleviating the aforementioned issues. However, we remained unsatisfied with this approach, as it essentially tripled the parameter count and removed possibilities for generalization between the 3 similar tasks.

In response to the shortcomings of `1-bert` and `3-bert`, we created a new model that incorporated what we saw as the best characteristics of each, which we termed `hybrid-bert`. We hypothesized that a disproportionate amount of the advantage of `3-bert` over `1-bert` was caused by the later layers of the transformers as well as the final pooling layers. Since all three tasks ultimately depend on similar features of natural language, it seemed unlikely that vastly different embedding and early transformer layers would be needed for them. However, since they each involve subtly different properties of sentences, it seemed likely that allowing the final layers of the BERT transformer to vary for each task would be advantageous. Accordingly, `hybrid-bert` used shared weights for the embedding and early transformer layers while allowing the final layers to specialize for each task.

We expected `hybrid-bert` to strike a balance in performance between `1-bert` (parameter-light, acceptable performance) and `3-bert` (parameter-heavy, best performance). So, we were pleasantly surprised when it matched the performance of `3-bert` with half as many parameters.

While the performance of `hybrid-bert` was the most exciting of our results, we performed a large variety of other experiments, many of which improved performance across the board. Both `1-bert` and `hybrid-bert` are based on multitask fine-tuning, which we implemented via a unified weighted loss function for overall performance, similar to the approach taken by Bi et al. (2022). We attempted to further optimize multitask training with gradient surgery per Yu et al. (2020a). We experimented with different loss functions, learning rate schedules, and dropout configurations. Some of our most substantial gains came from experimenting with different features for the multi-sentence tasks, as well as different architectures of the classification heads. Finally, we performed a manual hyperparameter sweep.

3 Related Work

Several authors have improved BERT since it was introduced in 2018. Liu et al. (2019) conducted a replication study of BERT pretraining and found that BERT was significantly undertrained. They introduced RoBERTa, A Robustly Optimized BERT Pretraining Approach, where they trained the model longer, on more data, and with bigger batch sizes. They also dynamically changed the masking pattern applied to training data. Sanh et al. (2019) adapted BERT for transfer learning, where operating large pretrained models on edge or with limited computational or inference budgets remains challenging. They propose DistilBERT, leveraging knowledge distillation during the pretraining phase to reduce the size of the BERT model by 40%, while increasing speed and retaining language understanding abilities. Lan et al. (2019) introduced ALBERT to increase the training speed of BERT through parameter reduction.

While these papers demonstrate effective ways to speed up BERT training on a single task by sharing weights within the transformer, we considered the parallel case of weight-sharing strategies between multiple transformers trained on different tasks. However, we found the success of techniques like ALBERT within a single transformer encouraging. In our paper, we encountered two main challenges when optimizing a multitask model: gradient interference and parameter optimization.

Gradient interference makes it difficult to infer which direction to move in to minimize multitask loss. To combat gradient interference, Yu et al. (2020b) propose "Gradient Surgery for Multitask Learning." They project a task's gradient onto the normal plane of another task that has a conflicting gradient. Therefore, the optimizer can move in one consistent direction. While gradient surgery was originally proposed for multitask reinforcement learning problems, Yu et al. (2020b) claimed that it was model agnostic. Therefore, we explore the effectiveness of gradient surgery in multitask learning with the BERT architecture. See an example of the process to transform conflicting into non-conflicting gradients in Figure 1.

BERT and PALs, by Stickland and Murray (2019), implemented parameter sharing in multitask learning. They realized that multitask learning involves conserved functionality between related tasks, which sometimes reduces the number of parameters required for the final model. Using projected attention layers (PALs), they augment the BERT architecture for more efficient parameter sharing by learning task-specific projections. These task-specific layers are interleaved in the main BERT layers, which are shared between all tasks. This improves the model's ability to generalize across multiple tasks with fewer parameters. Our project with `hybrid-bert` was similar to Stickland and Murray (2019), in that it identified a path to accelerate training by sharing parameters across BERT models.

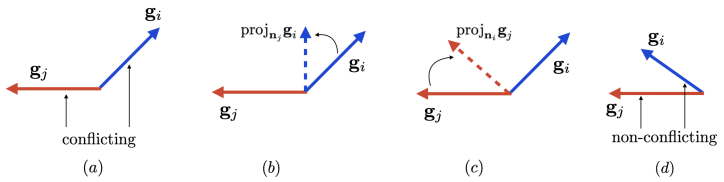

Figure 1: Conflicting gradients and PCGrad

However, we took a different approach, sharing and dividing BERT layers themselves rather than augmenting them with PAL layers.

4 Approach

4.1 Baseline and Early Approaches

Our baseline approach, described in Staff (2024), is passing frozen BERT embedding into task-specific linear layers that are trained for SST (per part 1 of the Default Project) and randomized for the remaining tasks.

The above-described baseline achieves poor performance on the Paraphrase and STS tasks since it is not trained on these tasks and merely uses the initialized weights. Our next step was to extend the training code to train on all three objectives. We applied binary cross-entropy loss for paraphrase detection and mean squared error loss for STS. All reported models were trained end-to-end, allowing the BERT weights to be tuned in addition to the classification heads.

Our first approach to the multitask training problem was to use these task-specific loss functions to train repeatedly our model (same architecture as baseline) for one epoch on each of the three tasks in succession. We experimented with learning rates, dropout rates, weight decay, and learning rate schedulers. We settled on a default learning rate of 1e-5 and a dropout rate of 0.3. We found that the OneCycleLR scheduler improved sentiment accuracy but harmed paraphrase accuracy and similarity correlation. Furthermore, it performed only slightly better than the CosineAnnealingLR scheduler. Weight decay was ineffective for similarity correlation but improved paraphrase accuracy. Ultimately, we came up with a more effective way to perform multitask training, described in the `1-bert` section, but these early experiments informed our choices with later models.

4.2 Gradient Surgery, Feature Augmentation, and Architectural Improvements

In all 3 of our final models, we implemented a variety of techniques to improve performance. Gradient surgery, described in Yu et al. (2020a), was used in `1-bert` and `hybrid-bert` to ameliorate the problem of conflicting gradients during multitask training. We used PCGrad, the implementation of Gradient Surgery in the original paper, as a starting point and integrated it into our code using the AdamW optimizer and our three task-specific loss functions. Qualitatively, gradient surgery mildly slowed convergence but led to more stable accuracy once convergence was reached. Gradient surgery did not, however, lead to a noticeable improvement in accuracy.

To add expressiveness at the (task-specific) classification steps, the linear classification heads were replaced by 3-layer feed-forward networks with ReLU activation after each layer and an intermediate layer width of 256[1]. We found shallower classification heads to lead to faster convergence at the expense of accuracy and deeper heads to result in overfitting, as indicated by falling dev-set accuracy while training loss continued to decline. For all tasks, a 3-layer network proved to be an effective compromise.

Initially, we used the BERT transformer to produce embeddings of the two input sentences and fed those concatenated embeddings into our classification head. We suspected that our low performance on sentence-pair tasks was due to the difficulty of our shallow classification heads in learning a similarity measure between the two embedding vectors, so we added the Hadamard product of the

[1]The width and depth of these networks was determined via hyperparameter sweep.

sentence embeddings as an additional input of the classification heads. We hoped that the heads would then learn to use a weighted dot-product of the embeddings as a more expressive version of cosine similarity, combined with information from the two embeddings themselves. Adding this feature substantially improved performance on the sentence-pair tasks. However, the best performance was attained by concatenating paired sentences with a separation token and feeding the combined result into BERT, as was recommended by Devlin et al. (2018).

4.2.1 `1-bert`: Unified Multitask Training

In response to the divided model's underwhelming performance on sentiment classification (compared with the end-to-end classifier from Part 1), we made major changes, motivated by our interpretation that training embeddings to simultaneously perform well on multiple tasks was degrading their performance on individual tasks. To prevent large gradient updates that advantaged one task at the expense of another, we rewrote our training loop to consider 3-tuples of batches from the 3 tasks at a time using a loss function calculated as a weighted sum of the 3 task loss functions. We implemented the changes described in the preceding section with the intention of further reducing the possibility of the three training objectives interfering. Since the different datasets had wildly different sizes, random sampling across epochs was required to preserve parity between the different tasks. The architecture of our final `1-bert` model is shown in Figure 2.

4.3 `3-bert`: Single Task Training

To determine whether our improvements had fully remedied the problem of conflicting optimization objectives, we tested an alternative approach, `3-bert` (compare in Figure 2). Instead of using the same transformer for all three tasks with multitask fine-tuning, we created 3 identical copies of the BERT transformer arranged in an ensemble, with one BERT instance dedicated to each task. Then, we adapted our training techniques from `1-bert` to perform simultaneous single-task finetuning of each instance. As is detailed in our experiments section, `3-bert` outperformed `1-bert` overall, indicating that the additional flexibility gained by allowing each transformer instance to specialize to a task was valuable. However, this came at the cost of tripling the parameter count and removing opportunities for generalization between tasks.

4.4 `hybrid-bert`: Layer Sharing with Multitask Training

Our final model, `hybrid-bert`, addressed the shortcomings of `1-bert` and `3-bert` with a novel layer-sharing scheme. The `hybrid-bert` architecture shares embedding layers and the first 6 out of 12 BERT layers between all three downstream tasks. However, the final 6 BERT layers and the pooling layers are duplicated, allowing for task-specific specialization. This architecture is contrasted with `1-bert` and `3-bert` in Figure 2. Our choice to share the early layers rather than the final ones was motivated by our hypothesis that optimal weights in early layers were likely to be conserved for all 3 similar tasks, while optimal weights in final layers were likely to be task-specific. We implemented `hybrid-bert` by adapting the original miniBERT architecture to set its early layers by reference to a shared object. Finally, the entire architecture was subjected to multitask fine-tuning, with all of the previously described improvements (gradient surgery, feature augmentation, etc.). We believed that `hybrid-bert` would be valuable even if it did not match the performance of `3-bert` due to requiring substantially fewer parameters. In fact, we were surprised by the very competitive performance of this model, which is described in the Experiments section.

5 Experiments

5.1 Data

BERT is fine-tuned and evaluated on the Stanford Sentiment Treebank SST dataset for sentiment analysis, the Quora dataset Quo for paraphrase detection, and the SemEval STS dataset Androutsopoulos et al. (2013) for semantic similarity tasks. The SST dataset consists of single sentences with a sentiment label of negative, somewhat negative, neutral, somewhat positive, or positive. The Quora dataset classifies sentence pairs as paraphrases or not, and the STS dataset scores sentence pair similarity. See Table 2.

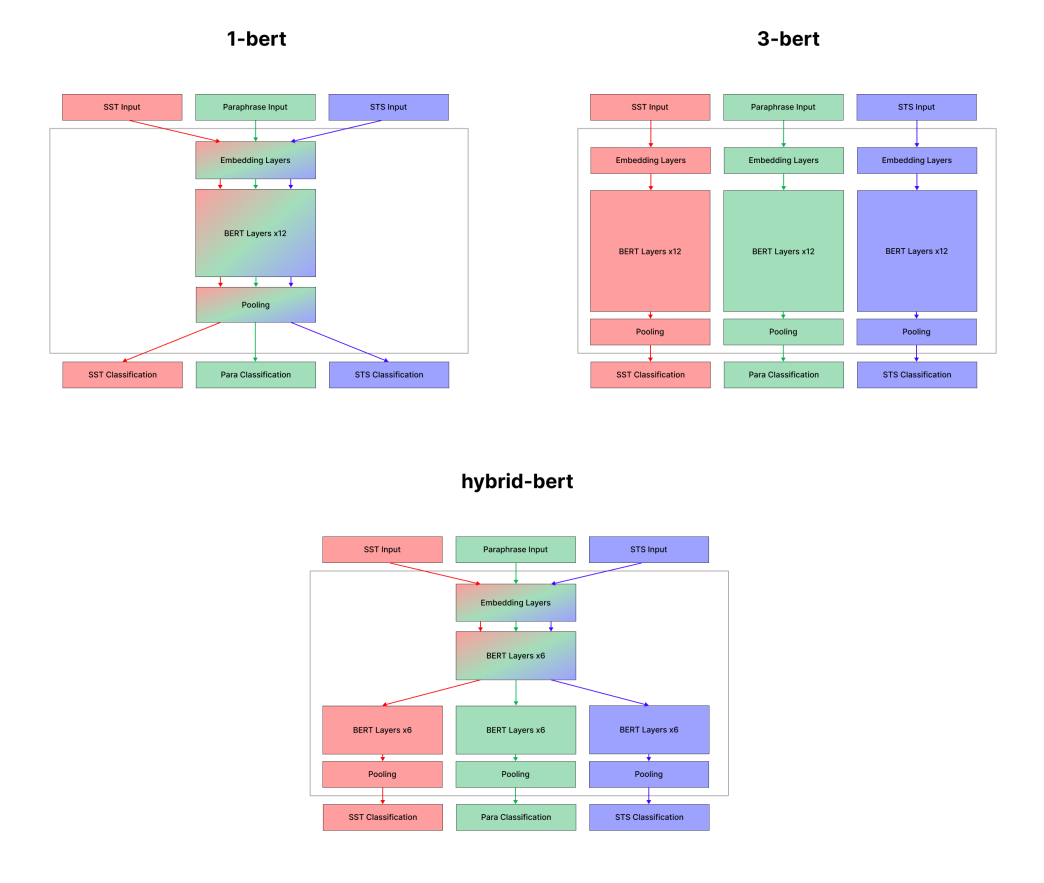

Figure 2: Schematic architectures of `1-bert`, `3-bert`, and `hybrid-bert`

5.2 Evaluation method

To evaluate BERT's performance, we used accuracy across all tasks. This was determined by the number of correct classifications using the dev set. We switched to the test set for the final results of our best model. For semantic similarity, we used the Pearson correlation coefficient. These metrics are defined in Staff (2024).

5.3 Experimental details

We trained the `1-bert`, `3-bert`, and `hybrid-bert` models for 20 epochs and evaluated them on the dev set. The results from our initial hyperparameter, learning rate scheduler, and weight decay sweep for the baseline model are reported in Figure 3 in the appendix. Ultimately, the model performed best with a weight decay of 0.001 and without a learning rate scheduler (linear or cosine). We also added dropout within the feedforward classification heads in addition to it initially being before the classification heads. We found that varying dropout beyond 0.3 did not result in a meaningful improvement in performance. For our final submission, we evaluated our best-performing model, `hybrid-bert`, on the test set. We trained `hybrid-bert` for 40 epochs with a learning rate of 1e-5, dropout of 0.3, weight decay of 0.001, and no learning rate scheduler. Training time took approximately 60 minutes on an NVIDIA A100.

5.4 Results

Our baseline model had around poor results overall. Similarity correlation especially underperformed because it wasn't trained on similarity-specific data. `1-Bert` performed significantly better than

the baseline, and 3-bert performed the best on the dev set. Surprisingly, hybrid-bert performed as well as 3-bert on the dev set, with an overall score of 0.772. hybrid-bert performed even better on the test set with scores reported in Table 1. This confirms that layer sharing is an incredibly effective method for reducing parameters while maintaining the performance of multitask models.

Model	Sentiment Acc.	Paraphrase Acc.	Similarity Corr.	Overall Score
Baseline (dev)	0.312	0.369	-0.009	0.392
1-bert (dev)	0.497	0.854	0.863	0.761
3-bert (dev)	0.513	0.868	0.872	0.772
hybrid-bert (dev)	0.510	0.871	0.869	0.772
hybrid-bert (test)	**0.526**	**0.882**	**0.879**	**0.782**

Table 1: Results for our final models

6 Analysis

To better understand the training dynamics of our three final models, we consider the multitask loss curves for their final (20-epoch) training runs presented in 3. All three models show asymptotic convergence to approximately 0 training loss within 20 epochs. While this is difficult to discern with a linear scale, the hybrid-bert and 3-bert models exhibit faster loss convergence than does 1-bert. We suspect that the additional task-specific expressiveness of these models enabled faster convergence on individual tasks, in addition to better performance evaluated on the dev set. For further breakdowns of our training curves, see figures 4, 6, and 5 in the Appendix.

We also considered stability during training. We noticed the unified multitask model 1-bert was more stable than 3-bert. hybrid-bert was also more stable than 3-bert, comparable to 1-bert. Furthermore, we noticed that although gradient surgery did not improve accuracy, it increased stability after convergence.

To better consider the shortcomings of our system, we performed qualitative error analysis by selecting poorly performing examples from the dev set for our final model. On the sentiment classification task, our final hybrid model achieved the worst performance on the following sentence from the dev set: "In a way, the film feels like a breath of fresh air, but only to those that allow it in." Our model predicted a sentiment of 1 (very negative), while the actual sentiment was 4 (positive). In fact, this error seems quite reasonable to us, and we expect that many humans would make a similar mistake. The sentence could easily be perceived as either critical or congratulatory, and determining which would likely depend on the surrounding context. However, rating it as the most negative sentiment category was still probably a case of overconfidence on behalf of the model.

For the sake of brevity, similar "worst-case" examples are included in the appendix for the other two tasks. It is worth noting that they are also marginal cases that would be confusing to human evaluators, too. The paraphrase detection example in particular seems questionably labeled; we disagreed with the evaluation included in the dev set and agreed with our model's prediction.

Given these worst-performing examples, we suspect that while there is still likely room for our model to improve, it came fairly close to approximating the performance of a human evaluator. Much of the remaining error may in fact be due to the subjective nature of labelling natural language tasks. Putting this aside, our hybrid-bert model exhibits the worst performance on sentences with unclear or debatable meaning, which is not particularly surprising.

7 Conclusions

In sum, we experimented with BERT's architecture, hyperparameters, data, loss function, additional adjustments like gradient surgery, and layer sharing. We found that changes in hyperparameters (including learning rate schedulers) resulted in minimal improvements in performance. Changing the dropout probability made a marginal difference in accuracy, but adding additional dropout layers within the feedforward classification heads was more impactful. Learning rate schedulers had mixed performance, and a small amount of weight decay was helpful. New data was significantly helpful in

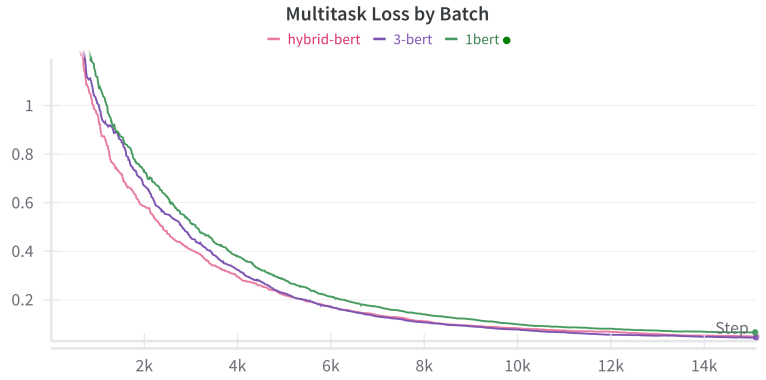

Figure 3: Multitask Loss Training Curves for our Final Models

increasing STS correlation, along with how data was concatenated. Adding the Hadamard product of sentence embeddings as an additional classification head input significantly improved performance on sentence-pair tasks.

Gradient surgery was marginally helpful in baseline models. It slowed convergence and led to more stable accuracy but did not increase accuracy once convergence was reached. It was not helpful in our final hybrid model. Unified loss, where we considered 3-tuples of batches to avoid moving too far in either the SST, STS, or paraphrase direction during descent, made a significant difference in overall performance. Finally, we found that shallower classification heads led to faster convergence but lower accuracy compared to deeper heads, which overfit. This suggested that we needed a balance between deep and shallow head size, which was one of the more important architectural choices that impacted overall accuracy.

Because we were hoping for more substantial increases in accuracy, we implemented `3-bert`, which was trained on each task individually. `3-bert` outperformed `1-bert`. However, it tripled parameter size and increased training time. This led to our novel contribution of a layer-sharing architecture `hybrid-bert`. `hybrid-bert` shares 6 embedding layers, but the final 6 layers are duplicated to allow for task-specific specialization. Surprisingly, `hybrid-bert` had identical accuracy to `3-bert`. In some cases, `hybrid-bert` outperformed `3-bert`, but we attribute this to model variance.

One limitation of our work was that our ultimate hyperparameter sweep was not as robust as we would have hoped. Another limitation was that although we implemented gradient surgery, we still encountered gradient interference during multitask training. This was especially prevalent in the earlier stages. This suggests that gradient surgery may lose effectiveness with deeper, more robust models. It would be worth exploring what models and loss functions gradient surgery works best on. Furthermore, there was an imbalance of data. The Quora dataset had significantly more examples than the other datasets combined. If there is bias in the Quora dataset or if the Quora dataset is best suited to certain tasks, this will propagate in our results. A naive extension of this paper would be to significantly increase training data and ensure limited bias.

We believe layer sharing to be an incredibly effective technique for maintaining multitask model performance while reducing parameters. This has important downstream applications where computational constraints must be considered. We suggest layer sharing be experimented on a wider variety of architectures such as RoBERTa and DistilBERT. We also suggest an ablation study where accuracy is recorded while altering the number of layers shared. Altering layer sharing may change fundamental model performance and require a reconsideration of other layers like dropout and head size. Future work may also explore how layer-sharing BERTs benefit from different types of data augmentation and dynamic task weighting.

8 Ethics Statement

8.1 Bias

Since a central feature of this paper is the production of semantic embeddings, a major concern is the potential for biases included in the training data to be included in the word embeddings themselves. This would be particularly concerning if the embeddings were later used for downstream tasks, as it could result in societal biases being perpetuated. One way to partially remedy this is to carefully select our training data to eliminate examples that could introduce bias. This could be done with a method as simple as a wordlist to screen training examples for harmful content, but it could also be more sophisticated. For instance, we could pass all training examples through an existing LLM to identify any examples that encode biases and prune them from the training set.

The multitask nature of our hybrid model introduces additional potential bias issues. Since weights are shared between three tasks, we cannot rule out the possibility that biases in one dataset might filter into outputs for the other tasks, which would exacerbate the harm caused by such biases. Ultimately, it is impossible to do justice to the ethical concerns raised by a language model without better understanding the areas to which it might be applied, but we would recommend a thorough ethical review before deployment with attention to these concerns.

8.2 Privacy

Another ethical concern is that this model could be used in a way that violates individual privacy. Specifically, semantic embeddings could be used by a downstream company to unethically profile users based on their comments and messages, such as for personalized advertising. This is a difficult issue to forestall entirely, as once a model is released, it may fall into the hands of nefarious users. However, one possible approach would be to intentionally limit the usefulness of the embeddings when the input text appears sensitive. Training examples could be provided of phrases that indicate the model is being inappropriately applied to user data, such as texting abbreviations. Then, an additional fine-tuning pass could be used with a destructive loss function to ensure, for instance, that these inputs result in nonsensical embeddings while normal inputs are unaffected. This approach may reduce the utility of the embedding model to bad actors.

References

First quora dataset release: Question pairs. `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`. Accessed: 2024-06-07.

Stanford sentiment treebank (sst). `https://nlp.stanford.edu/sentiment/treebank.html`. Accessed: 2024-06-07.

Ion Androutsopoulos, Prodromos Malakasiotis, John Pavlopoulos, and Fotini Voyatzi. 2013. Semeval-2013 task 4: Free paraphrases of noun compounds. In *Second Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 14–22.

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

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

CS224n Staff. 2024. Default final project handout. Accessed: 2024-05-22.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. *CoRR*, abs/1902.02671.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020a. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020b. Gradient surgery for multi-task learning. *CoRR*, abs/2001.06782.

A Appendix

This appendix provides information on data and our baseline multitask model with various learning rate schedulers and weight decays.

The following are the examples for which our predictions were the worst for the STS and Paraphrase tasks. If multiple examples were tied for worst, we included only one, which was randomly selected.

- **Paraphrase Detection**
  - S1: "Why computer vision is hard?"
  - S2: "Why computer vision is computationally hard?"
  - Predicted score: 1
  - Actual score: 0
- **Semantic Textual Similarity**
  - S1: "Work into it slowly."
  - S2: "It seems to work."
  - Predicted score: 3.2
  - Actual score: 0

Name	Train	Dev	Test
SST	8,544	1,101	2,210
CFIMDB	1,701	245	488
Quora	141,506	20,215	40,431
SemEval STS	6,041	864	1,726

Table 2: Datasets

Model	Sentiment Acc.	Paraphrase Acc.	Similarity Corr.
OneCycleLR, No Weight Decay	**0.506**	0.701	0.337
CosineAnnealingLR, No Weight Decay	0.505	0.714	0.367
CosineAnnealingLR, 0.001 Weight Decay	0.505	0.714	0.367
No Scheduler, 0.0001 Weight Decay	0.464	0.676	**0.404**
No Scheduler, 0.001 Weight Decay	0.464	0.676	**0.404**
No Scheduler, 0.01 Weight Decay	0.467	**0.726**	0.370
No Scheduler, No Weight Decay	0.464	0.676	**0.404**

Table 3: Accuracy of the baseline multitask model with various learning rate schedules and weight decays

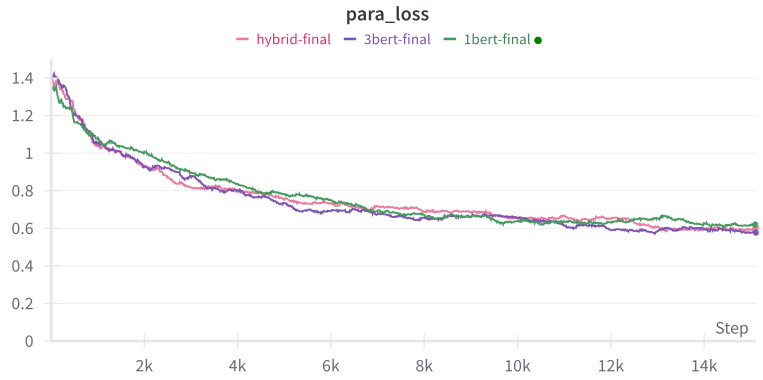

Figure 4: Final Training Curves: Paraphrase Detection

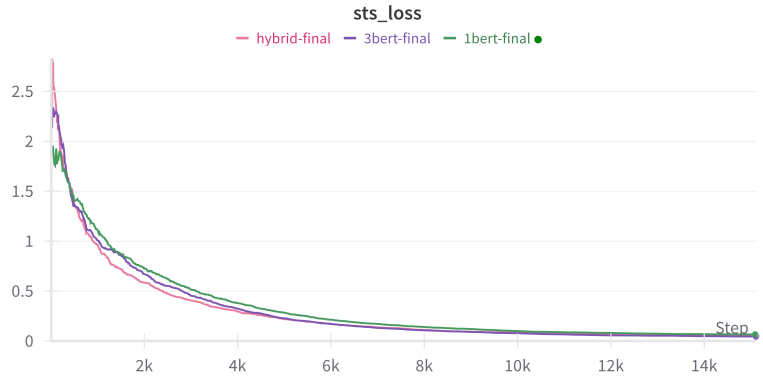

Figure 5: Final Training Curves: STS

Figure 6: Final Training Curves: SST