# Finetuning minBERT for Downstream Tasks with Multitasking

Stanford CS224N default Project

**Niall Kehoe**
Department of Computer Science
Stanford University
nkehoe@stanford.edu

**Pranav Ravella**
Department of Computer Science
Stanford University
pravella@stanford.edu

## Abstract

Creating fine-tuned language models of pre-trained transformer language models are extremely powerful tools that have made an immense revolution in the field of machine learning. However, fine-tuning these models on single down stream tasks do not generalize across various other tasks. We want to extend the Bidirectional Encoder Representations Transformers (minBERT) model to make high quality predictions on multiple sentence-level tasks, namely sentiment analysis, paraphrase detection, and semantic textual similarity. The advantage of using one model with a multitasking capability is it creates more robust and generalized sentence embeddings which perform well on a variety of tasks. We experiment with various method to create generalized embeddings extending off the minBert model with techniques and avoiding task interference as much as possible, such as Gradient Surgery (Yu et al., 2020), Gradient Vaccine (Wang et al., 2020), SMART regularization techniques for fine-tuning (Jiang et al., 2019), model embedding optimizations, and stronger multi-head networks

## 1 Introduction

The landscape of natural language processing (NLP) has experienced a notable shift with the emergence of pre-trained language models like minBERT. However, a limitation of these models is their inability to perform the best multiple tasks. For instance, fine-tuning them for a single specific task often hampers their ability to generalize across other tasks. To address this issue, multi-task learning has emerged as a promising strategy. This approach enables models to undergo training on multiple tasks simultaneously, facilitating the exchange of knowledge between tasks and leading to enhanced efficiency, generalization, and overall performance. In this experiment, we understand how minBERT can be optimized on three different NLP downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Some of these multi-tasking optimizations included Gradient Surgery (Yu et al., 2020), Gradient Vaccine (Wang et al., 2020), SMART regularization techniques for fine-tuning (Jiang et al., 2019), model embedding optimizations, and stronger multi-head networks.

To train our model, we utilized the Stanford Sentiment Treebank dataset (SST) (Socher et al., 2013), the Quora dataset for paraphrase detection (Iyer et al., 2012), and the SemEval STS Benchmark dataset (STS) (Agirre et al., 2013) for semantic textual similarity as the various tasks. Our findings indicate that multitask fine-tuning with gradient vaccine, contextual embeddings in paraphrase and semantic textual similarity detection tasks, and SMART regularization was the best method of creating a model that compromise on all three tasks the best.

## 2 Related Work

When training for mutliple downstream tasks the gradients for each task can often conflict, making it more difficult to converge. Yu et al. (2020) presents a possible solution (PCGrad) to this issue, by presenting an approach for gradient surgery for multitask learning. This involves projecting a task's

gradient onto the normal plane of the other tasks which have conflicting gradients. This technique has been show in other multi-task applications to effective at improving convergence and performance.

Wang et al. (2020) also presents an alternative solution to the same problem of conflicting gradients in multitasking models called Gradient Vaccine. Gradient Vaccine solves some of the problems faced by PCGrad, one being the assumption that all tasks have similar gradient interactions. Gradient vaccine works by changing the magnitude and direction of the gradients so that some gradient similarity requirement is met.

Jiang et al. (2019) tackles the issue of over fitting with aggressive fine-tuning leading to poor performance on unseen data. Jiang et al. (2019) tackles this by incorporating a regularization function designed induce smoothing (reducing complexity/variability of decision boundaries) which improves generalization. It also adds Bregman proximal point optimization which regularizes so that the parameter updates during finetuning are not too aggressive. This makes for a smoother and more stable convergence during training.

# 3 Approach

## 3.1 Implementation of minBERT and Baseline Model

We implemented the minBERT model, where we completed the multi-head attention layer of the BERT transformer (Devlin et al., 2018) and the AdamW optimizer (Kingma and Ba, 2017). We added 3 different heads for each respective task. Using our minBERT model, we pre-trained and fine-tuned using just for each dataset to achieve a baseline on the task. For SST, the head takes the embeddings from BERT, applies dropout, and then a linear layer that reduces the output to the number of sentiment classes. For paraphrase and STS tasks, the embeddings of each of the two input sentences after dropout is computed and combined together. Then, this is put through another linear layer for the final outputs. For STS, the linear layer output dimension is equal to 5 or the number of classes and the prediction is the largest outputted logit with a basic cross entropy loss function. For paraphrase detection a binary logit is outputted and a binary cross entropy loss function is utilized. We treat the SST task a regression method and use an mean-squared-error loss function from the final output of the linear layer.

$$\hat{y}_{sentiment} = Linear(Dropout(BERT_{output}))$$

$$\hat{y}_{sts} = Linear(Dropout(BERT^{(1)}_{output}), Dropout(BERT^{(2)}_{output}))$$

$$\hat{y}_{paraphrase} = Linear(Dropout(BERT^{(1)}_{output}), Dropout(BERT^{(2)}_{output}))$$

## 3.2 Sequential Multi-task Fine-tuning

Our multi-tasking baseline model used a multi-task approach for the three tasks aforementioned. In every epoch, the model goes through all the batches for a single dataset and backpropagates appropriately through all the batches. It then goes through the other two datasets in a similar manner within the same epoch. The loss functions for each task is independent and not combined. This was purely a sequential approach to give us a baseline of what training on all three datasets would result (Algorithm 1).

## 3.3 Multi-task Fine-tuning with Round Robin Scheduling

Multi-tasking with the sequential method did not accurately capture how the different tasks should be combined with each other when backpropogating through the model and updating the weights. As a result, we utilized a round-robin approach to update the model with the knowledge of all three tasks at once. For every batch index, we would predict each task and calculate the loss function. Then, a gradient treatment would be applied to the three loss functions and an update would occur. In this approach, we had to align the number of batches we could cycle through, since the number of STS training examples was much smaller than the paraphrase training dataset. As a result, we aligned the batch index to go up to the end of the STS dataset and then randomly sampled without replacement for all of the datasets (Algorithm 2). This meant that the model effectively underfit for the paraphrase dataset and slightly for SST dataset. It is important to note that the indexes for the SST and paraphrase datasets have a saved state iterator that cycled once it reached the end of the dataset. So after 1.5 epochs, the SST batch iterator would restart.

**Algorithm 1** Multi-Tasking Baseline Model Training

**procedure** TRAINMULTITASK($model, num\_epochs, SST\_dataset, STS\_dataset, Para\_dataset$)
    Initialize model parameters $\theta$
    Initialize optimizer $opt$
    **for** $epoch \leftarrow 1$ to $num\_epochs$ **do**
        TRAINONDATASET($model, SST\_dataset, opt, sst\_loss$)
        TRAINONDATASET($model, STS\_dataset, opt, sts\_loss$)
        TRAINONDATASET($model, Para\_dataset, opt, paraphrase\_loss$)
    **end for**
**end procedure**
**procedure** TRAINONDATASET($model, dataset, optimizer, loss\_fn$)
    **for** $batch \in dataset$ **do**
        $logits \leftarrow model(batch\_input, task)$
        $loss \leftarrow loss\_fn(logits, batch\_labels)$
        $loss.backward()$
        $optimizer.step()$
        $optimizer.zero\_grad()$
    **end for**
**end procedure**

**Algorithm 2** Pseudocode for Multi-Task Round-Robin Model

$batch\_indices \leftarrow range(len(sts\_dataset))$
**for** $epoch$ in $num\_epochs$ **do**
    **for** $batch\_idx$ in $batch\_indices$ **do**
        $sst\_batch \leftarrow sst\_dataset[sst\_batch\_idx]$
        $para\_batch \leftarrow para\_dataset[para\_batch\_idx]$
        $sts\_batch \leftarrow sts\_dataset[batch\_idx]$
        $sst\_outputs \leftarrow model(sst\_batch)$
        $para\_outputs \leftarrow model(para\_batch)$
        $sts\_outputs \leftarrow model(sts\_batch)$
        $sst\_loss \leftarrow loss\_function(sst\_outputs, sst\_labels)$
        $para\_loss \leftarrow loss\_function(para\_outputs, para\_labels)$
        $sts\_loss \leftarrow loss\_function(sts\_outputs, sts\_labels)$
        $total\_loss \leftarrow sst\_loss + para\_loss + sts\_loss$
        $total\_loss.backward()$
        $optimizer.step()$
        $optimizer.zero\_grad()$
    **end for**
**end for**
**Loss**
$\mathcal{L} = \mathcal{L}_{SST} + \mathcal{L}_{Para} + \mathcal{L}_{STS}$

### 3.3.1 Gradient Surgery

In order to mitigate the issue of conflicting gradients in multi-task fine-tuning, we implemented gradeint surgery using PCGrad (Yu et al., 2020), a method to mitigate opposing gradients from impairing training. By projecting the task-specific gradients onto the normal plane of the gradient vector sum and then consolidating them, PCGrad aims to reduce the interference between conflicting gradients while still allowing the model to leverage the shared representations across tasks. This equation represents the projection of the gradient vector $\vec{g}_i$ of the $i$-th task onto the normal plane of another conflicting task's gradient vector $\vec{g}_j$.

$$\vec{g}_i = \vec{g}_i - \frac{\vec{g}_i \cdot \vec{g}_j}{\|\vec{g}_j\|^2} \vec{g}_j$$

After having $[\mathcal{L}_{SST}, \mathcal{L}_{Para}, \mathcal{L}_{STS}]$, PCGrad was then used appropriately with its optimizer.

### 3.3.2 Gradient Vaccine and Gradient Accumulation

The paraphrase task and STS are more similar compared to the SST task, and such should not be treated with equal similarity to one another. Gradient vaccine (Wang et al., 2020) seeks too modify the gradient of task $i$ by adding a component that is proportional to the gradient of task $j$, scaled by a factor that depends on the angle between the two gradients and the target angle $\phi_{ij}^T$. Gradient vaccine

encourages the gradients of different tasks to become more aligned and allows inter-task relationships to be more evident during training.

$$\vec{g}_i' = \vec{g}_i + \frac{\|\vec{g}_i\|}{\|\vec{g}_j\|}\left(\phi_{ij}^T - \phi_{ij}^2 - \phi_{ij}\sqrt{1-(\phi_{ij}^T)^2}\right)\vec{g}_j$$

Furthermore, batch sizes of more than 16 caused the model to run out of memory. As a result, we utilized gradient accumulations to create a virtual batch size of 32 to improve over-fitting risk. Gradient accumulations allowed gradient updates to be taken at delayed staged, while still summing the batches' gradients up to that point.

### 3.4 Comparison Contextual Embeddings

The traditional approach of encoding both pairs of sentences with BERT for the Paraphrase and STS tasks, and then running a classifier on the two outputs of the encodings showed weak performance. Standalone sentences have less context than comparing them side by side as humans do.In order to simulate this behavior we concatenated the two sentences together and marked the break between the two using separator tokens. This created one bigger sentence which we then fed into BERT to get the embeddings. One large text embedding gives the model a greater view of the two sentence context's allowing it to make smarter decisions over their similarity / differences.

### 3.5 Regularized Optimization

With fine-tuning our models with multi-task learning and the aforementioned gradient treatments, the model still had a risk of over-fitting on the data. As a result, we utilized the SMART regularization framework Jiang et al. (2019) to diminish this risk and improve the models ability to generalize on data.

The SMART frameworks reduces over-fitting and generalization of limited data with two steps. The Smoothness-Inducing Adversarial Regularization step

$$\min_{\theta} \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_{\mathrm{s}}\mathcal{R}_{\mathrm{s}}(\theta)$$

$$\mathcal{R}_{\mathrm{S}}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\max_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} \ell_{\mathrm{S}}\left(f\left(\widetilde{x}_i;\theta\right), f\left(x_i;\theta\right)\right),$$

$\mathcal{L}(\theta)$ is the loss function after gradient treatment, $\lambda_{\mathrm{s}} > 0$ is a tuning parameter, and $\mathcal{R}_{\mathrm{s}}(\theta)$ is the regularizer. By incorporating this regularization technique, it acts as a constraint that discourages extreme updates to the model's parameters, ensuring that the model learns more stable and generalizable patterns from the data.

The second step is bregman proximal point optimization.

$$\theta_{t+1} = argmin_{\theta}\mathcal{F}(\theta) + \mu\mathcal{D}_{Breg}\left(\theta, \theta_t\right)$$

where $\mu > 0$ is a tuning parameter, and $\mathcal{D}_{Breg}(\cdot,\cdot)$ is the Bregman divergence defined as

$$\mathcal{D}_{Breg}\left(\theta, \theta_t\right) = \frac{1}{n}\sum_{i=1}^{n}\ell_{\mathrm{s}}\left(f\left(x_i;\theta\right), f\left(x_i;\theta_t\right)\right)$$

This prevents abrupt changes to the paramaters during training. It prevents $\theta_{t+1}$ update to be signifcant of that to $\theta_t$ update.

#### 3.5.1 Fine-tuning with Initial Paraphrase Task

Training the multi-task learning system for 5 epochs on just the paraphrase task before the rest of the tasks allowed the model to start at a point where it was guided towards higher scores for STS and paraphrase tasks. We saw general improved performance from just training on the paraphrase dataset across all three tasks, and it increased accuracy by 0.1 across all tasks. This was only implemented for the models with SMART and gradient vaccine combined.

### 3.6 Bigger Networks in task-specifc Heads

For each head, we explored adding a few hidden layers in order to allow it to model more complicated relationships between the BERT outputs and the downstream output. This helped improve performance slightly but required longer training periods, a larger number of epochs, and caused memory constraint issues. We ended up using a hidden layer of size BERT_HIDDEN_SIZE for the SST head, a 2 deep hidden layer with size BERT_HIDDEN_SIZE and BERT_HIDDEN_SIZE//4 respectively for both Paraphrase and STS.

# 4 Experiments

## 4.1 Data

Our pre-trained model is a BERT-base-uncased model (Devlin et al., 2018). This pre-trained model uses a masked language modeling (MLM) method. Furthermore, the pre-trained model has around 110 million paramaters trained a diverse and large English language dataset. For our downstream tasks, we fine-tuned the model on 3 datasets.

**1.** We used the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013) for the sentiment classification task, where the given text is classfied by its polarity with 5 classes. The dataset has 11,855 sentences that were extracted from movie reviews. Furthermore, these sentences were split into 215,154 phrases and labeled by human judges in one of the 5 classes (0 is the most negative and 4 is the most positive).

**2.** For the paraphrase detection task, we employed the Quora dataset (Iyer et al., 2012). This dataset contained 400,00 lines of questions that could be duplicates of each other. The input is a pair of questions and the output is either 0 or 1 for not being a paraphrases of each other or not.

**3.** For the Semantic Text Similarity (STS) task we utilized the SemEval STS Benchmark dataset (Agirre et al., 2013). This dataset had 8,628 different sentence pairs. The dataset aimed to measure the semantic degree of pairs of sentence from 0 (dissimilar) to 5 (basically equivalent).

## 4.2 Evaluation method

For the SST and Quora dataset based tasks, we measure the accuracy of the model on the test and dev datasets. Accuracy is measured by the number of examples that are correctly classified by the model in the set proportion to the total number of examples in the set. For the SemEval dataset task, we evaluate the model's performance using the Pearson Correlation score (Agirre et al., 2013). This score measures the linear relationship between the predicted and actual scores, which is a proper evaluation metric for continuous values.

## 4.3 Experimental details

We were able to emulate a higher batch size with gradient accumulations that started after implementing gradient vaccine. Adjusting to the higher batch size, we also increased the learning rate and epochs to get better results. Furthermore, we only fine-tuned the minBERT model and no use of the frozen pre-trained minBERT model was done (Table 1).

Table 1: Model Configurations

| Model | Configuration | Batch Size | Epochs | Learning Rate |
|---|---|---|---|---|
| Sequential Multitask learning | | 16 | 10 | $1 \times 10^{-5}$ |
| Gradient Surgery + CON | | 16 | 10 | $1 \times 10^{-5}$ |
| Gradient Vaccine + RR | | 32 | 15 | $1 \times 10^{-5}$ |
| Gradient Surgery + RR | | 16 | 15 | $1 \times 10^{-5}$ |
| Gradient Surgery + RR + CON + HLH | | 16 | 15 | $2 \times 10^{-5}$ |
| Gradient Vaccine + RR + CON + HLH | | 16 | 15 | $2 \times 10^{-5}$ |
| SMART + GVAC + RR + CON | | 32 | 15 | $2 \times 10^{-5}$ |
| SMART + GVAC + RR + CON + HLH | | 32 | 15 | $3 \times 10^{-5}$ |

### 4.4 Results

Table 2: Dev Set Model Performances

| Model | epochs | SST Acc. | Para Acc. | STS Cor. | Overall Score |
|---|---|---|---|---|---|
| SST head fine tuning only | 10 | 0.52 | 0.43 | 0.057 | 0.0493 |
| Sequential Multitask learning | 5 | 0.501 | 0.741 | 0.353 | 0.64 |
| Individual Pre-training and Finetuning | 5 | 0.521 | 0.762 | 0.384 | 0.658 |
| Gradient Surgery + CON | 10 | 0.503 | 0.812 | 0.863 | 0.749 |
| Gradient Vaccine + RR | 15 | 0.480 | 0.757 | 0.375 | 0.642 |
| Gradient Surgery + RR | 15 | 0.453 | 0.748 | 0.364 | 0.628 |
| Gradient Surgery + RR + CON + HLH | 15 | 0.509 | 0.826 | 0.857 | 0.755 |
| Gradient Vaccine + RR + CON + HLH | 15 | 0.494 | 0.858 | 0.865 | 0.762 |
| SMART + GVAC + RR + CON | 15 | **0.521** | 0.854 | 0.887 | 0.772 |
| SMART + GVAC + RR + CON + HLH | 15 | 0.517 | **0.871** | **0.891** | **0.778** |

RR: Round Robin, GVAC: Gradient Vaccine, CON: Comparison Contextual Embedding, HLH: Hidden Layer Heads

Table 3: Test Set Model Performances

| Model | epochs | SST Acc. | Para Acc. | STS Cor. | Overall Score |
|---|---|---|---|---|---|
| Gradient Surgery + RR | 15 | 0.512 | 0.841 | 0.861 | 0.761 |
| SMART + GVAC + RR + CON + HLH | 15 | **0.516** | **0.870** | **0.884** | **0.776** |

RR: Round Robin, GVAC: Gradient Vaccine, CON: Comparison Contextual Embedding, HLH: Hidden Layer Heads

- **Sequential Multitask Learning**: The results for the baseline sequential model performs better than the model with gradient surgery and round-robin, but around the same as the model with gradient vaccine and round-robin. This may be because it synthesizes common information from across the datasets. However, we see through training that the conflicting gradients causes the model to over fit for the paraphrase task, since it has a abundance of data compared to the other tasks. Furthermore, the model performs better than we expected it to, but the comparison is hard to make as the models with gradient treatments was trained on a dataset aligned to the STS dataset size. As a result, the paraphrase information that could help the tasks was not used.
- **Gradient Surgery + CON**: This model greatly improved performance over previous methods, seeing a massive boost in the STS Correlation (0.353 -> 0.863). It also increased Para Acc. significantly (0.741 -> 0.812). By combining the embeddings of the two sentences into a single tensor it allows the model to gain a more effective understanding of both. Examining both sentences leads to a more complete understanding the measure of semantic textual similarity as well as identifying paraphrases.
  Gradient surgery also helped improve performance by improving the optimization process as we have tasks which vary in difficulty (illustrated by the gap in performance across categories)
- **Gradient Vaccine + RR**: This model outperformed the gradient surgery configuration, which was what we initially predicted. It helped increase the STS score as well, which was less aligned to the other two tasks and is the purpose of using cosine similarity when calculating the gradient to project on.
- **Gradient Surgery + RR**: In comparison to the sequential model, we initially predicted that the model would outperform it. However, as seen from the results, the lack of data that it was trained on showed in the results. As a result, even with the limited data (1/10th of sequential), it performed similarly, which demonstrates how good gradient surgery is as removing the conflicting gradients. This model configuration aligned to the paraphrase dataset and cycling the other two would have made the model overfit on the other two tasks and took around 8 hours to complete just 6 epochs. As a result, when testing, this model could not be use but should be looked into for future research.

- **Gradient Surgery + RR + CON + HLH**: By combining the Round Robin approach with the Comparison Contextual Embeddings we see a slight boost in performance over Gradient Surgery + CON. We also added 1 hidden layer for each task's head which might have allowed the model to make more complex connections.
- **Gradient Vaccine + RR + CON + HLH**: Again, as with Gradient surgery, combining previous successful methods we see a performance improvement.
- **SMART + GVAC + RR + CON**: The SMART regularization, along with fine-tuning the model with the paraphrase dataset to start, made slight improvements to the contextual embedding model on all tasks, and this is directly due to SMART allowing for the model not to over-fit on the training data, as well as getting a push in the right direction with the STS and paraphrase tasks from the initial fine-tune. This was what initially predicted when implementing SMART, however, we hoped for a bigger jump in benchmarks.
- **SMART + GVAC + RR + CON + HLH**: By adding hidden layers to each task head, more complex relationships can be captured and this is why we see a slight boost in performance.

## 5 Analysis

**SST**:
We see our model is often off in our prediction by 1, but much less likely to be off by 2 or 3 (Figure 1).
**Incorrect**:
An example which was labelled a 4, but we predicted a 1 for is *"the acting , costumes , music , cinematography and sound are all astounding given the production 's austere locales ."* This sentence use complex vocabulary that we suspect is not seen in the training set "austere locales". This makes it difficult to predict as the model is unsure of the meaning of this words. An example which was labelled a 0 but we predicted a 3 for is *"it 's everything you do n't go to the movies for."* We believe here the space between the "do" and "n't" instead of "don't" confused the model into predicting a positive but was actually negative.
**Correct**:
Example which were labelled a 4 and we predicted correctly is *"a warm , funny ,engaging film"*. This review is short and direct which is much easier to predict. *"dazzles with its fully-written characters , its determined stylishness -lrb- which always relates to characters and story -rrb- and johnny dankworth 's best soundtrack in years ."* was also correct. These reviews vary in length but are both straightforward and direct in their review, making it easier to predict by the model.

**Para**:
We see our model is more likely to make the mistake of falsely predicting a paraphrase when none exits (false positive) than it is to not see a paraphrase when one is present (false negative) (Figure 2).
**Incorrect**:
An example which was not a paraphrase, but we predicted to be was *"what 's your favourite jackie chan movie ?"* and *"what are the best movies of jackie chan ?"*. This sentence uses a lot of the same words, which make it similar but the model fails to spot the difference in singular/plural. An example which was a paraphrase but we predicted it was not was *"why does india have immigration checks/passport control while leaving the country ?"* and *"what does immigration officers check on their screen while leaving india ?"*. These sentences use many different words to express the same meaning, with different sentence structure which we believe confused the model as they appeared to be very different on the surface despite capturing the same underlying meaning.
**Correct**:
An example we correctly classified was *"how do i speak english fluently ?"* vs *"how can i speak english more fluently ?"*. This example is relatively simple as there is only one word that is different between them.

**STS**:
Graphing a histogram of our STS errors, we see it is approximately normal but is slightly positively centered, meaning it is more likely to underpredict the STS score (Figure 3).
**Incorrect**:
An example which was similar (3.2), but was predicted to be (0.111): *"in these days of googling , it 's sloppy to not find the source of a quotation ."* and *"i agree with kate sherwood , you should be*

*able to attribute most quotes these days by simple fact checking ."* Despite these quotes feeling very disconnected to the model, there is an underlying theme of searching for quotations that it missed. We think one reason for this was the use of "quotation" in one and "quote" in the other.

**Correct**:
An example in which our error was 0.0005 (true value of 0.2, prediction was 0.1995) was *"two lambs stand on a grassy hill ."* and *"two dirt bikers riding over dirt hill ."* Our model correctly spotted these two things had very few things in common appear from "two" and "lambs". One of the reasons the error is so low was that sentence was simple and used the same vocab for similar parts but very different vocab for the unconnected portions.

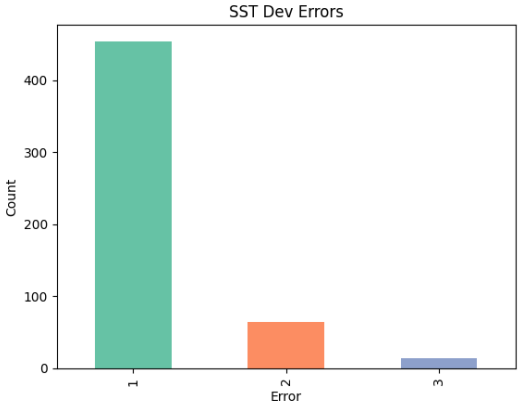


Figure 1: STS Dev Errors


Figure 2: Para Error

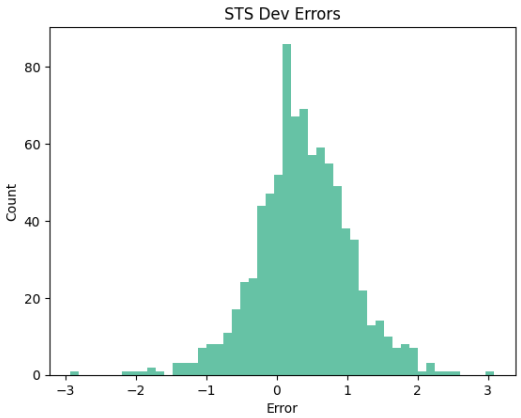


Figure 3: STS Error

## 6 Conclusions

Through trail and error we found certain techniques that were effective on this multitasking objective and others that proved less worthwhile. Comparison Contextual Embedding (CON) led to a major improvement in performance for both the Para and STS tasks as they involved comparison between two sources of text, which each contained less context to base decisions of individually than when they were combined together. Round Robbin (RR) led to a slight boost in performance across all metrics. We found Gradient Vaccine (GVAC) led to better scores than Gradient Surgery in general. Combining SMART, Gradient Vaccine, Round Robbin, Comparison Contextual Embeddings and Hidden Layer Heads for downstream tasks lead to the best model performance in Para, STS and second best in SST. Further work may improve upon this by incorporating pre-training for each of the three task heads and further enhancing pretraining for the upstream minBERT model.

## References


Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

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.

Shankar Iyer, Nikhil Dandekar, and Csernai Kornél. 2012. First quora dataset release: Question pairs. Quora.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. SMART: robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *CoRR*, abs/1911.03437.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. 2020. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. *CoRR*, abs/2010.05874.

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