Parameter-Efficient Learning Strategies for Multi-Task Applications of BERT

Stanford CS224N Default Project

Irmak Sivgin
Electrical Engineering, Stanford University
isivgin@stanford.edu

Mahmut Yurt
Electrical Engineering, Stanford University
myurt@stanford.edu

Abstract

In this project, we employ multiple strategies to improve the downstream performance of BERT in sentiment classification, paraphrase detection, and semantic similarity scoring tasks. Our goal is to fine-tune the base BERT model with task-specific downstream layers to produce robust embeddings generalizable to multiple tasks. We approach this problem by investigating various multi-task learning improvements (multi-stream BERT, BERT with cosine-similarity), by exploring optimization techniques (gradient surgery), by incorporating parameter-efficient learning strategies (LoRA, DoRA, rs-LoRA), and by utilizing different token embeddings (using the [CLS] token, average of all hidden features, [CLS] token and averaged tokens processed and concatenated). We find that the best performance is achieved by multi-stream BERT with cosine similarity, using both averaged and concatenated features optimized by gradient surgery on the multi-task objective.

1 Key Information to include

- **TA mentor:** Tony Lee. **External collaborators, external mentor, sharing project:** No.

2 Introduction

Natural language processing (NLP) is a vital tool in modern computer science for enabling machines to comprehend, interpret, and interact with the human language Nadkarni et al. (2011). NLP can tackle a wide range of diverse tasks, including but not limited to text classification Kowsari et al. (2019), machine translation Macherey et al. (2001), and information retrieval Voorhees (1999). For sentence-level analysis, NLP excels in handling medium-length texts, providing precise and accurate results. In particular, NLP is shown to be highly effective in sentiment analysis to detect the emotional tone in a given text, paraphrase detection to determine if two sentences convey similar meanings, and semantic textual similarity regression to evaluate the degree of similarity between sentences. However, the field still faces important challenges such as ambiguous language, contextual complexity, and idiomatic expression Khurana et al. (2023).

Building generalizable language models is therefore essential for robust techniques that can perform well across diverse domains and tasks Cheng et al. (2018). Multi-task learning (MTL) is a promising approach to achieve generalizability by training a single model across multiple tasks simultaneously, leveraging shared representations to enhance performance Caruana (1997). For instance, widely-used models like bidirectional encoder representations from transformers (BERT) have demonstrated the efficacy of shared representations in obtaining high performance across various NLP tasks Devlin et al. (2018). However, MTL significantly increases the number of parameters to be optimized, resulting in a training process with higher computational requirements. Furthermore, balancing the optimization of unique task-specific features with shared features is quite challenging. Note that task-specific features are crucial for fine-tuning the model to perform well at individual tasks, whereas shared features provide a robust foundation across all tasks Yurt et al. (2021). Meanwhile, task interference, in which learning one of the tasks negatively impacts another one, also complicates the balance in multi-task learning Yu et al. (2020). The challenge of managing these aspects highlights the difficulty of developing generalizable models in NLP, necessitating techniques like parameter-efficient learning, adaptive parameter sharing and task-specific regularization to navigate these challenges effectively.

In this project, we aim to address the challenges of multi-task learning using a pre-trained BERT model for sentiment analysis, paraphrase detection, and semantic similarity regression. For this purpose, we demonstrated a multi-stream approach to effectively learn both task-specific and shared representations. This was achieved using a network architecture that pools information from task-specific linear layers and a shared linear layer used across tasks. To address task interference, we explored gradient surgery, which involves projecting a gradient onto the orthogonal space of another gradient in cases of gradient conflicts Yu et al. (2020). To enhance performance, we further utilized a cosine similarity metric in semantic similarity regression. Moreover, we explored parameter-efficient learning strategies such as LoRA Hu et al. (2021), DoRA Liu et al. (2024), and rs-LoRA Kalajdzievski (2023) to manage the increased number of parameters in BERT needing optimization. These techniques introduced different low-rank adaptations, where the weights of the pre-trained BERT are frozen and trainable low-rank matrices are added. Finally, we examined fusing additional hidden states from BERT, including direct use of the pooling output, averaging, and concatenation.

3 Related Work

Multi-task learning: The idea of multi-task learning aims to enhance overall model performance by simultaneously handling multiple tasks Caruana (1997); Yurt et al. (2021), and has been widely explored in the context of language models Collobert and Weston (2008); Radford et al. (2019); Chen et al. (2023). Among these works, Collobert and Weston (2008) trained a joint deep neural network across various tasks and showed enhanced generalizability. Similarly, Radford et al. (2019) suggested that a language model can benefit from large, diverse datasets, in an unsupervised learning framework for performance benefits over multiple domains. More recently, Chen et al. (2023) introduced a multimodal large language model as a unified interface for various vision-language multi-task learning applications. This is achieved by proposing unique identifiers for individual tasks during learning Chen et al. (2023). There is another stream of research on improving optimization on multi-task objective landscape. In Bi et al. (2022), authors simply add all the individual losses from different tasks and perform backpropagation on this merged objective. Although this idea can be powerful, authors of Yu et al. (2020) note the possibility of detrimental gradient interference among different task objectives, proposing "gradient surgery", a procedure that projects a gradient onto the other gradient's normal plane for in the case the gradients conflict. This approach of modifying gradients for multi-task learning has been shown to improve efficiency and performance. In a recent paper, Yurt et al. (2021) demonstrated that pooling information from unique and shared features for multiple tasks can effectively enhance performance. This is achieved by learning individual and shared layers separately and then concatenating them to learn a final fusion layer Yurt et al. (2021).

Parameter-efficient learning: One limitation of fine-tuning large language models is its need for substantial computational resources to handle the complexity of the networks Hadi et al. (2023). To address this various strategies have been developed Houlsby et al. (2019); Hu et al. (2021); Liu et al. (2024). Among them, a recent study introduced low-rank adaptation (LoRA) that freezes the weights of the pre-trained model and injects trainable rank decomposition matrices Hu et al. (2021). This helps to significantly reduce the number of trainable parameters and computational cost Hu et al. (2021). Building on this approach, another study proposed a corrected scaling factor for rank stabilized LoRA, namely rsLoRA Kalajdzievski (2023). Meanwhile, another study proposed weight-decomposed low-rank adaptation, DoRA, that decomposes the pre-trained weight matrices into magnitude and directional components, and fine-tunes them Liu et al. (2024). These methods also address the issue of overfitting to the smaller fine-tuning dataset and drifting away from the pretrained backbone model, reducing the number of trainable parameters in the fine-tuning process.

4 Approach

Here we describe the baselines and main approaches used in our study to enhance performance.

4.1 Baselines

We considered two baselines (see Appendix Fig. 1). The first one involves multi-task training of BERT Devlin et al. (2019) for sentiment analysis, paraphrase detection, and semantic textual similarity, with a single additional linear layer dedicated to each task, where only the additional linear layers are trained, and BERT is not fine-tuned. The second baseline utilizes the same architecture and multi-task training; however, here BERT is also fine-tuned to examine if it improves the performance. We refer to this as the base model in the remainder of the report.

4.2 Multi-Stream Bert

As an architectural contribution to multi-task learning, we first developed a multi-stream approach (MS) inspired by Yurt et al. (2021). This MS approach integrates a shared linear layer (SL) and three separate task-specific linear layers (TL) following BERT output (see Appendix Fig. 2). These shared and task-specific layers aim to leverage both shared and task-specific representations to enhance model performance for all tasks. The implementation can be summarized as follows:

- **BERT Encoder:** The given input text is provided to the pre-trained BERT model that outputs the contextualized token representations [CLS]. The token output denoted as $\mathbf{h}_{\text{CLS}}$ serves as the aggregate representation of the input sequence.
- **Shared Linear Layer (SL):** The [CLS] output $\mathbf{h}_{\text{CLS}}$ is processed by the shared linear layer to generate a general representation across tasks:

$$\mathbf{h}_{\text{shared}} = \mathbf{W}_{\text{shared}}\mathbf{h}_{\text{CLS}} + \mathbf{b}_{\text{shared}} \tag{1}$$

  where $\mathbf{W}_{\text{shared}}$ denotes the weights, $\mathbf{b}_{\text{shared}}$ the biases, and $\mathbf{h}_{\text{shared}}$ the output of the shared layer.
- **Task-Specific Linear Layers (TL):** Depending on the ongoing task, the same [CLS] output $\mathbf{h}_{\text{CLS}}$ is propagated through the corresponding task-specific linear layer:

$$\mathbf{h}_{\text{task}_i} = \mathbf{W}_{\text{task}_i}\mathbf{h}_{\text{CLS}} + \mathbf{b}_{\text{task}_i} \tag{2}$$

  where $\mathbf{W}_{\text{task}_i}$ denotes the weights, $\mathbf{b}_{\text{task}_i}$ the biases, and $\mathbf{h}_{\text{task}_i}$ output of the task-specific linear layer for the ith task.
- **Concatenation:** The outputs of the shared and task-specific linear layers are concatenated to form a combined representation:

$$\mathbf{h}_{\text{combined}_i} = [\mathbf{h}_{\text{shared}}; \mathbf{h}_{\text{task}_i}] \tag{3}$$

  where $[;]$ denotes the concatenation operation.
- **Subsequent Layers:** The combined representation $\mathbf{h}_{\text{combined}_i}$ is propagated to the next layer to compute the final output for the respective task.

$$\mathbf{h}_{\text{out}_i} = \mathbf{W}_{\text{last}_i}\mathbf{h}_{\text{combined}_i} + \mathbf{b}_{\text{last}_i} \tag{4}$$

  where $\mathbf{W}_{\text{last}_i}$ denotes the weights, $\mathbf{b}_{\text{last}_i}$ biases, and $\mathbf{h}_{\text{out}_i}$ output of the final layer.

4.3 Cosine Similarity

Cosine similarity is a common metric used in NLP due to its ability to capture orientation rather than magnitude of the vectors. Therefore, we utilized it for the task of semantic textual similarity to evaluate the degree of the likeness between the two token representations Reimers and Gurevych (2019). Its formulation is given as:

$$\cos\text{similarity}(x_1, x_2) = \frac{{x_1}^\top x_2}{max(||x_1||||x_2||, \epsilon)} \tag{5}$$

where x_1 and x_2 are the token representations, and ϵ is a small value preventing division by zero. The output ranges in [-1, 1], where 1 indicates that vectors are identical in direction, 0 indicates that vectors are orthogonal, -1 indicates that vectors are diametrically opposed. Here the output is propagated through a ReLU followed by a multiplication by 5 to perform the desired regression.

4.4 Multitask Fine-Tuning (PCGrad)

In multitask optimization, it is possible to have gradients that have negative conflict with other task gradients, which triggers over- or under-estimation problems Yu et al. (2020). Authors propose projecting conflicting gradients (PCGrad) method to remove the conflicting components of the gradients across tasks in Yu et al. (2020). The notion of gradient conflict is defined as i-th task gradient g_i and j-th task gradient g_j having a negative dot product, i.e., ${g_i}^\top g_j < 0$. In such cases, g_i is modified as: $g_i \leftarrow g_i - \frac{{g_i}^\top g_j}{{g_j}^\top g_j} g_j$ Yu et al. (2020). After this gradient modification, the model parameters are optimized through regular gradient descent. For this part, AdamW optimizer is used. We base our implementation of PCGrad on the original GitHub code, but adapt to PyTorch instead [1].

[1] `https://github.com/tianheyu927/PCGrad`

4.5 Parameter Efficient Fine-Tuning

Low-rank adaptation for fine tuning (LoRA), as introduced in Hu et al. (2021), alleviates the memory and computation overhead in the fine tuning procedure by freezing the pretrained parameters and only making low-rank updates. For any targeted layer that will be fine-tuned, the weights are re-parameterized as $W_0 + AB$ with the pretrained model weight $W_0 \in \mathbb{R}^{d\times k}$ being untouched, and $A \in \mathbb{R}^{d\times r}, B \in \mathbb{R}^{r\times k}$ as the parameters to be updated Hu et al. (2021). The rank of the modification, $\Delta W = AB$ is $r \ll d, k$, which enables parameter efficiency in fine tuning while also preventing overfitting to the specific task. LoRA utilizes a scaling parameter α such that ΔW is scaled by $\frac{\alpha}{r}$, which the authors fix at a scalar in the order of r as optimizing α is roughly the same as optimizing the step size Hu et al. (2021). Thus, the final update rule for LoRA is

$$\hat{W} = W_0 + \frac{\alpha}{r}\Delta W = W_0 + \frac{\alpha}{r}AB. \tag{6}$$

In a recent study, authors show that dividing the scale factor by $\sqrt{r}$ provides rank-stabilization, which enables improved performance at higher values of r, while original LoRA performance does not improve much with increasing r Kalajdzievski (2023). Rank-stabilized LoRA (rs-LoRA) thus provides for a fine-tuning compute/performance trade-off.

Weight-decomposed low-rank adaptation (DoRA) decomposes the pretrained model weight W_0 to magnitude and direction and fine-tunes both components, where the direction parameters are adapted with the LoRA method Liu et al. (2024). The authors claim that concentrating LoRA on the directional component of the weights simplifies the task, as the magnitude component is fine-tuned separately, and that this approach stabilizes optimization. For a weight matrix $W \in \mathbb{R}^{d\times k}$, the magnitude-direction decomposition is expressed as

$$W = \frac{V}{\|V\|_c}m = \frac{W}{\|W\|_c}\|W\|_c \tag{7}$$

where $m \in \mathbb{R}^k$ is the magnitude vector holding the magnitude of each column of W, $V \in \mathbb{R}^{d\times k}$ is the directional matrix, and $\|\cdot\|_c$ denotes the norm of each column. We express DoRA fine-tuning:

$$\hat{W} = \frac{V + \Delta V}{\|V + \Delta V\|_c}m = \frac{W_0 + AB}{\|W_0 + AB\|_c}m \tag{8}$$

with $m = \|W_0\|_c, V = W_0$ at initialization. m is kept trainable while V is frozen, forcing low-rank updates on the direction matrix through LoRA procedure Liu et al. (2024). We utilized peft library from HuggingFace to implement LoRA, rs-LoRA and DoRA.

4.6 Token Fusing Strategies

To enhance model performance, we can leverage additional hidden states from BERT instead of relying only on the [CLS] token embedding. For this purpose, we examined two other token fusing strategies in addition to utilized the [CLS] token on its own:

- **[CLS] Token Embedding:** The [CLS] token embedding $\mathbf{h}_{\text{CLS}}$ is received from BERT with a size of (d_{hidden}), the hidden size of the BERT model.
- **Averaging:** The averaging approach involves calculation of the average of all hidden states, described as:

  $$\mathbf{h}_{\text{avg}} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{h}_i \tag{9}$$

  where L is the sequence length, $\mathbf{h}_i$ represents the hidden state at position i, and $\mathbf{h}_{\text{avg}}$ denotes the output with the same shape as the [CLS] token embedding, (d_{hidden}).
- **Concatenation with Non-Linear Transformation:** To pool information, the [CLS] token embedding $\mathbf{h}_{\text{CLS}}$ is concatenated with the non-linearly processed, averaged hidden states $\mathbf{h}_{\text{avg}}$. The process can be formulated as:

  $$\mathbf{h}_{\text{tra}} = \tanh(\mathbf{W}_{\text{avg}}\mathbf{h}_{\text{avg}} + \mathbf{b}_{\text{avg}}) \tag{10}$$

  $$\mathbf{h}_{\text{cat}} = [\mathbf{h}_{\text{tra}}; \mathbf{h}_{\text{CLS}}] \tag{11}$$

where $\mathbf{W}_{\text{avg}}$ represents the weights, $\mathbf{b}_{\text{avg}}$ denotes the biases, and $\mathbf{h}_{\text{tra}}$ is the output of the non-linear transformation, with a shape of (d_{model}), $\mathbf{h}_{\text{cat}}$ denotes the concatenation of the [CLS] token embedding and the linearly processed output of the averaged embeddings, resulting in a shape of $(2d_{\text{hidden}})$.

These token fusing strategies aim to leverage more comprehensive contextual information from BERT's hidden states, thereby improving the model's performance on the tasks.

5 Experiments

5.1 Data

Three different datasets are used: 1) SST dataset, 2) Quora dataset, and 3) SemEval STS Benchmark dataset. SST dataset includes 8,544 train, 1,101 dev and 2,210 test samples with annotated sentiment classification labels for 5 classes. Quora dataset consists of 283,010 train, 40,429 dev, and 80,859 test samples. The task is binary paraphrase detection. STS dataset consists of 6,040 train, 863 validation, and 1,725 test samples. The task is similarity value regression for given two texts. To mitigate misbalances between the datasets in the multi-task learning protocol, around 20,000 samples are randomly selected and used in training during each epoch for the Quora dataset.

5.2 Evaluation method

For the SST dataset that provides labels for a 5-class classification task, we evaluated performance using accuracy. Similarly, for the Quora dataset, where the task is to determine if particular instances are paraphrases, we also used accuracy in evaluation. The accuracy can be defined as: $\text{Accuracy} = (\text{Number of Correct Predictions})/(\text{Total Number of Predictions})$. Meanwhile, for the SemEval dataset, the task is a regression problem to predict the similarity value between pairs of texts. Here, to evaluate the performance, we use the Pearson correlation coefficient (PCC). PCC measures the linear correlation between the predicted vs. actual similarity values, and it is defined as: $r = (n(\sum y\hat{y}) - (\sum y)(\sum \hat{y}))/\sqrt{[n\sum y^2 - (\sum y)^2][n\sum \hat{y}^2 - (\sum \hat{y})^2]}$ where n is the number of paired scores, and y and $\hat{y}$ represent the actual and predicted similarity values, respectively. This metric helps us understand how well the predicted values match the true similarity scores.

5.3 Experimental details

Our experiments span architectural changes, utilizing different input features, parameter-efficient learning and multi-task optimization with gradient surgery. In terms of different models that are trained, we have the base model (MultiTaskBERT), base model augmented with shared layer (SL), multi-stream (MS) and cosine similarity (CS). For all these models, we perform parameter-efficient fine-tuning with LoRA, DoRA and rs-LoRA. The rank for the updates is set to 8, however, we also explore a higher rank (256) with rs-LoRA to see if the rank stabilization will boost the performance given more number of trainable parameters. The α parameter is set to 16.

We further explore the effect of using various features extracted from BERT in the downstream tasks by assessing performance of our models with the regular [CLS] token, averaging and concatenation strategies outlined in Section 4.6.

Finally, we select the best performing model-feature pairs to inspect the potential of PCGrad optimization. In order to handle the different dataset sizes across tasks, we randomly oversample from the smaller datasets to match the number of samples in the larger datasets. We observed that oversampling the smaller datasets allows for better PCGrad optimization compared to undersampling the larger datasets to match the size of the smallest dataset. Note that exact matching of the number of batches is necessary for the PCGrad procedure. PCGrad uses AdamW as the backbone optimizer as in all other cases. The learning rate is set to 1e-5. The number of training epochs is set to 10. All experiments were conducted on NVIDIA A40 Gpus, on Python using PyTorch.

5.4 Results

We present our results for multi-task learning with BERT. We report accuracy (sentiment classification and paraphrase detection) and correlation (similarity regression) on the dev set. Base model

(MultiTaskBERT) without any fine-tuning achieves 0.308 accuracy in sentiment classification, 0.634 accuracy in paraphrase detection and a correlation score of 0.225 in similarity evaluation, resulting in an average performance score of 0.389. In the tables that we present below, we include the fine-tuned BERT model as the comparison baseline, as fine-tuning the whole model improves upon the case where only the classification layers are trained. In the tables, SL refers to the base model with an additional shared layer whereas MS refers to the whole multi-stream model (see Section 4.2). CS stands for using cosine similarity for similarity evaluation task.

Tables 1, 2 and 3 display the performances of low-rank fine-tuning methods LoRA, DoRA and rs-LoRA, respectively. Inspecting the average scores across all models and fine-tuning methods with $r = 8$, the best performance is achieved by rs-LoRA and MS + CS model. Notably, each one of the three fine-tuning methods outperforms the other two for at least one model. Thus, we can say that these methods perform similarly. Testing rs-LoRA with $r = 256$ slightly increases the average scores, however, this difference was not observed to be significant.

We also present the affect of BERT embeddings on downstream task performances in Table 4. Token fusing by averaging or concatenation produces better results compared to only using the [CLS] token. This is an expected result as in either token fusing approach, the downstream model is still informed of the [CLS] token, but, also has access to the remaining hidden states of BERT, which convey more information. Interestingly, averaging seems to be more desirable for models without cosine similarity, and concatenation yields better results for models with cosine similarity. The best overall method appears to be MS + CS model with concatenated tokens. SL + CS and MS + CS models perform similarly, and both token fusing strategies yields good results. Thus, we compare naive SGD with PCGrad on these two models.

Table 5 displays SL + CS and MS + CS models with averaged or concatenated features optimized by naive AdamW or the PCGrad method. We observe that using gradient surgery improves results across all models and all token choices. We achieve the same combined score for MS + CS model with both averaged and concatenated features.

Finally, our test leaderboard results with MS + CS model using concatenated features optimized with gradient surgery are: **SST test accuracy: 0.505**, **Paraphrase test accuracy: 0.790**, **STS test correlation: 0.833**. Thus, we achieve the **overall test score** of **0.737**.

Model	Trainable %	Sentiment	Paraphrase	Similarity	Average Score
Base Model	1.478	0.417	0.730	0.254	0.467
SL	1.456	0.360	0.718	0.307	0.462
SL + CS	1.456	0.366	0.710	0.750	0.609
MS	1.458	0.322	0.723	0.301	0.449
MS + CS	1.458	0.322	0.717	0.697	0.579

Table 1: Comparison of LoRA fine-tuned model performances across the tasks.

Model	Trainable %	Sentiment	Paraphrase	Similarity	Average Score
Base Model	1.315	0.448	0.725	0.364	0.512
SL	1.294	0.395	0.719	0.357	0.491
SL + CS	1.294	0.362	0.679	0.749	0.597
MS	1.296	0.299	0.720	0.334	0.451
MS + CS	1.296	0.321	0.682	0.722	0.575

Table 2: Comparison of DoRA fine-tuned model performances across the tasks.

6 Analysis

We performed qualitative analysis to understand our model, and wanted to address when it works well and when it fails. In the following part, some examples are reported and commented on.

Sentiment classification:

- **Sentence:** A poignant, artfully crafted meditation on mortality. (4)
- **Prediction:** 4 – **Correct Answer:** 4

	Model	Trainable %	Sentiment	Paraphrase	Similarity	Average Score
$r = 8$	Base Model	1.478	0.420	0.714	0.238	0.457
	SL	1.456	0.361	0.730	0.278	0.456
	SL + CS	1.456	0.383	0.730	0.710	0.608
	MS	1.458	0.374	0.713	0.287	0.458
	MS + CS	1.458	0.415	0.727	0.719	0.620
$r = 256$	Base Model	32.24	0.423	0.730	0.327	0.493
	SL	32.07	0.441	0.723	0.293	0.486
	SL + CS	32.07	0.431	0.731	0.711	0.624
	MS	32.07	0.414	0.738	0.312	0.488
	MS + CS	32.07	0.421	0.707	0.701	0.610

Table 3: Comparison of rs-LoRA fine-tuned model performances across the tasks.

Model	Feature	Sentiment	Paraphrase	Similarity	Average Score
Base Model	[CLS]	0.511	0.726	0.367	0.535
	average	0.514	0.741	0.362	0.539
	concatenation	0.511	0.729	0.369	0.536
SL	[CLS]	0.510	0.737	0.367	0.538
	average	0.500	0.729	0.394	0.541
	concatenation	0.522	0.732	0.370	0.541
SL + CS	[CLS]	0.507	0.743	0.761	0.670
	average	0.523	0.745	0.805	0.691
	concatenation	0.523	0.747	0.805	0.692
MS	[CLS]	0.492	0.750	0.372	0.538
	average	0.497	0.751	0.371	0.540
	concatenation	0.511	0.744	0.355	0.537
MS + CS	[CLS]	0.491	0.745	0.797	0.678
	average	0.504	0.731	0.826	0.687
	concatenation	0.519	0.745	0.826	0.697

Table 4: Comparison of token fusing strategies with the choice of using the [CLS] token across the baseline and proposed models evaluated for 3 tasks.

For this example, we observed that our model is successful in predicting the sentiment of the given sentence. We can understand that the model correctly identified the positive sentiment conveyed by words like "poignant" and "artfully crafted", which indicate high praise and admiration.

- **Sentence:** This flick is about as cool and crowd-pleasing as a documentary can get.
- **Prediction:** 1 – **Correct Answer:** 4

The model might have struggled with understanding the context and the overall tone of the sentence, focusing on individual words such as flick, cool, crowd rather than the sentence as a whole.

Paraphrase detection:

- **Sentence 1:** How do I get that peace of mind?
- **Sentence 2:** What do you do to achieve peace of mind?
- **Prediction:** Paraphrase – **Correct Answer:** Paraphrase

The model is correct in this prediction. We sense that the content and intent of both sentences are the same, and we estimate that the model accurately identified the synonymous nature of "get" and "achieve", as well as the interchangeable structure of the questions. The model recognizes the equivalence of "how do I" and "what do you do to", which shows understanding of phrases.

- **Sentence 1:** Should I buy an iPhone 5s in 2016?
- **Sentence 2:** Is it advisable to buy iPhone 5s over a same priced Android smartphone in 2016 from a future perspective for at least 2 years?
- **Prediction:** Paraphrase – **Correct Answer:** Not Paraphrase

Model	Feature	Optimizer	Sentiment	Paraphrase	Similarity	Average Score
SL + CS	average	PCGrad	0.495	0.811	0.774	0.693
	concatenation	PCGrad	0.488	0.802	0.805	0.698
	average	AdamW	0.523	0.745	0.805	0.691
	concatenation	AdamW	0.523	0.747	0.805	0.692
MS + CS	average	PCGrad	0.489	0.816	0.835	0.713
	concatenation	PCGrad	0.496	0.807	0.837	0.713
	average	AdamW	0.504	0.731	0.826	0.687
	concatenation	AdamW	0.519	0.745	0.826	0.697

Table 5: Direct optimization vs. using PCGrad for the best performing models. We compare the performance of SL + CS and MS + CS models with enhanced features under different optimizations.

The model is incorrect in this prediction. We see that the two sentences are asking a similar question but in a much different manner. The first one directly ask the question on "iPhone", whereas the second one includes other pieces such "Android", "future". We estimate that the model was influenced by the shared context of purchasing decisions and technological considerations, resulting in not capturing the entire meaning.

Semantic similarity regression

- **Sentence 1:** Two girls playing with people and flowers behind them.
- **Sentence 2:** Two young girls wearing skirts are playing in a garden.
- **Prediction:** 3.287 – **Correct Answer:** 3.2

The model was almost perfect in predicting the similarity in this case as the two sentences are straightforward and quite similar. The model was able to match "flowers behind" with the concept of being "in a garden".

- **Sentence 1:** Queen pays tribute to Nelson Mandela.
- **Sentence 2:** South Africa's rugby fraternity mourns Mandela.
- **Prediction:** 3.01 – **Correct Answer:** 1

The model might lack the ability to deeply understand the context (Queen paying tribute vs rugby fraternity paying tribute), leading it to incorrectly assume a higher degree of similarity based on shared keywords rather than the overall meaning and specific entities involved.

7 Conclusion

In this project, we demonstrated a multi-stream approach for multi-task learning of BERT using parameter-efficient learning techniques, input fusing strategies, and gradient update techniques for sentiment classification, paraphrase detection, and semantic similarity regression. Using the proposed techniques, we achieved sentiment classification accuracy of 0.505, paraphrase test accuracy of 0.790, and similarity regression correlation of 0.833, yielding an overall test score of 0.737. The future work will be based on incorporation of external datasets to improve performance.

8 Ethics Statement

Developing a multi-task NLP model for sentiment analysis, paraphrase detection, and semantic similarity regression does not essentially seem dangerous, but there can be several ethical issues that one should address. Firstly, the risk of bias in the training data can lead to unfair or discriminatory outcomes. These outcomes can promote societal prejudices. Secondly, the use of personal data, especially in sentiment analysis, can raise privacy concerns. From individuals, proper consent might be necessary to collect, or anonymization should be performed. Additionally, the potential misuse of the model, such as manipulating opinions or conducting surveillance can pose ethical risks. To address these concerns, as developers, we should ensure that the training data we use is diverse and representative to minimize biases. We should ensure that we perform strong anonymization techniques to protect personal data. Furthermore, we should build transparent models with development and documentation to enhance accountability and trust.

References

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 of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28:41–75.

Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. 2023. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*.

Zhi-Qi Cheng, Xiao Wu, Siyu Huang, Jun-Xiu Li, Alexander G Hauptmann, and Qiang Peng. 2018. Learning to transfer: Generalizable attribute learning with multitask neural model search. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 90–98.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora.

Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. 2023. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information*, 10(4):150.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation.

Klaus Macherey, Franz Josef Och, Hermann Ney, et al. 2001. Natural language understanding using statistical machine translation. In *INTERSPEECH*, pages 2205–2208. Citeseer.

Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. 2011. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Ellen M Voorhees. 1999. Natural language processing and information retrieval. In *International summer school on information extraction*, pages 32–48. Springer.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning.

Mahmut Yurt, Salman UH Dar, Aykut Erdem, Erkut Erdem, Kader K Oguz, and Tolga Çukur. 2021. mustgan: multi-stream generative adversarial networks for mr image synthesis. *Medical Image Analysis*, 70:101944.

A Appendix (optional)

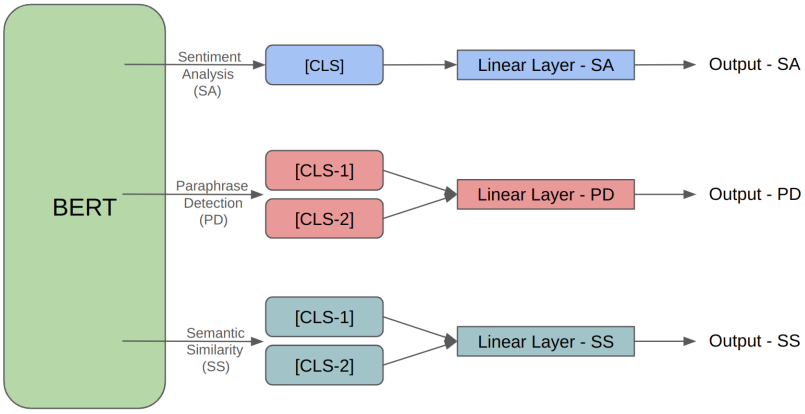

Figure A1: The network architecture for the base model used in the comparisons. The CLS token calculated for the sentiment analysis task is propagated through the linear layer to calculate output for that task. The [CLS] tokens for the two instances are calculated for the paraphrase detection, and they are concatenated before passing through the linear layer of that task. Similarly, the [CLS] tokens for the two instances are calculated for the semantic similarity task. They are again concatenated before passing the linear layer of the similarity task.

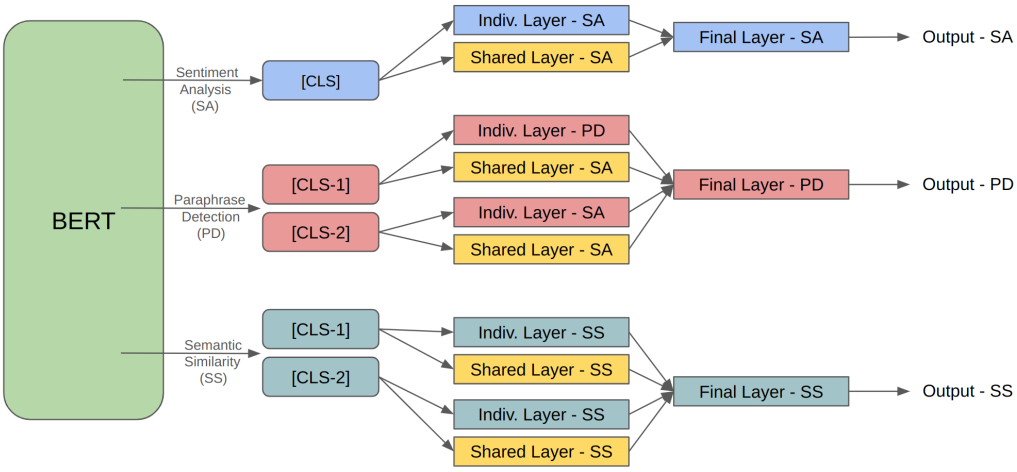

Figure A2: The multi-stream architecture is displayed. In this architecture, there is a shared layer shown in yellow here. Regardless of the task, [CLS] tokens are propagated through this layer to calculate representations. For the sentiment analysis task, [CLS] token is also propagated through the individual layer of this task, and the outputs from the individual and shared layers are concatenated before passing through the final layer for the sentiment analysis task. For the paraphrase detection, the two [CLS] tokens are separately propagated through the shared layer and individual layer of this task, and then all representations are concatenated before passing through the final layer of this task. Similarly, for semantic similarity, the two [CLS] tokens are passed through the individual and shared layers, and then the outputs of these layers are concatenated and passed through the final layer.