MT-DNN with SMART Regularisation and Task-Specific Head to Capture the Pairwise and Contextually Significant Words Interplay

Stanford CS224N Default Project

Haoyu Wang
Department of Computer Science
Stanford University
haowang3@stanford.edu

Abstract

In this study, we develop a novel BERT-small-based architecture tailored for multi-task fine-tuning on three downstream NLP tasks: sentiment analysis on the SST dataset, paraphrase detection on the Quora dataset, and evaluating semantic textual similarity on the STS Benchmark dataset. Our architecture proposes task-specific head enhancements, incorporating 1D convolutions and BiLSTM for sentiment classification, and a shared pairwise word interaction model (PWIM) for both sentence pair tasks, each followed by dedicated task-specific classifiers. We also introduce an innovative training paradigm to address disparities in dataset size scale across tasks. Initially, we employ the Smoothness-Induceing Adversarial Regularization technique (SMART) for further pre-training to refine our contextual embeddings. Subsequently, LoRA and adapters are integrated for efficient fine-tuning for each specific task. Notably, the Quora dataset exclusively further fine-tunes the shared PWIM layer, which is then frozen, followed by the STS task, which only fine-tunes its task-specific components. This multi-faceted approach shows our architecture's generalisation on multiple downstream tasks while navigating the challenges posed by varying dataset scales.

1 Introduction

In the evolving landscape of NLP, the paradigm of pre-training followed by fine-tuning on multiple downstream tasks has emerged as a standard for equipping models with the versatility to excel at diverse challenges simultaneously. The general pipeline typically involves further pre-training to enhance the contextual embeddings with task-specific datasets to tailor the model for particular applications, followed by dedicated fine-tuning for task-specific heads. However, regularization emerges as a critical concern during further pre-training, given the underlying BERT model is extremely complex while the limited datasets. So without careful regularisation and we further pre-train aggressively on these task datasets, it will lead to an overfitting contextual embedding, compromising its ability to generalize to new, unseen data and undermining the potential benefits of any subsequent task-specific adjustments.

When doing task-specific fine-tuning, for sentiment classification, capturing the nuanced interplay between contextually significant words is crucial for accurately gauging sentiment. Similarly, or tasks assessing sentence similarity, effective models mimic human cognition, comparing semantic similarities across sentences and identifying corresponding elements i.e. the model needs to have the capacity to handle the correspondence pairwise words regardless their actual positions in their sentences. Besides that, a significant challenge in this multi-task learning framework is the varying scale of datasets for different tasks. If we want to employ complex heads to capture these pairwise word interactions for each task, that means we necessitate substantial training data for effective generalisation. However, SST and STS datasets being relatively small (less than 10,000 samples)

and the Quora dataset significantly larger (over 1 million samples), posing a dilemma for employing complex interaction head for SST and STS, but can be trained for Quora task. It is noticeable that the Quora and STS tasks, while distinct in their binary versus regression nature, share underlying similarities that lend themselves to shared model architecture. By leveraging a shared pairwise word interaction layer for both tasks (PWIM), complemented by task-specific classifiers, we can harness the knowledge gained from the Quora task to enhance performance on the STS task, thus addressing the challenges posed by dataset scale disparities and task-specific requirements.

2 Related Work

2.1 Smoothness-Inducing Adversarial Regularization & Bregman Proximal Point Optimization (SMART)

SMART encompasses two regularization techniques; however, this study focuses solely on smoothness-inducing adversarial regularization and Jiang et al. (2019)'s original work demonstrates significant generalization improvements attributed primarily to this approach. By endorsing the concept of local Lipschitz continuity, this regularization technique ensures minimal changes in the function f output with the introduction of a small perturbation (by lp norm ϵ) to the input data. Illustrative examples in the paper depict the decision boundary in a 2D classification problem (Figure 1), with and without regularization, underlining its potential to mitigate overfitting during further pre-training on downstream task data.

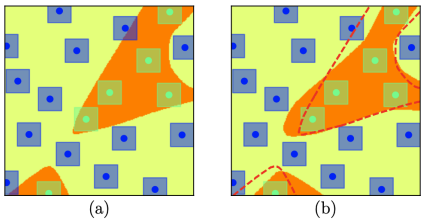

Figure 1: Decision boundary with and without SMART regularization.

2.2 1D Convolution Layer and Bi-LSTM for Sentiment Classification

Tam et al. (2021) introduces a pipeline incorporating a 1D convolution layer followed by a max-pooling layer and multi-layer Bi-LSTM to discern the intricate contextually significant words interplay (Figure 2), thereby enhancing sentence embedding. Drawing inspiration from this model, our implementation adapts the concept by employing a tailored 1D convolution and single-layer Bi-LSTM by considering the dataset size constraints, which will be discussed in detail later.

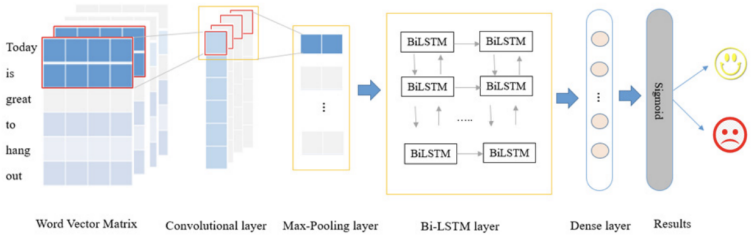

Figure 2: 1D convolution layer followed by Bi-LSTM architecture.

2.3 Pairwise Word Interaction Model (PWIM)

Proposed by He and Lin (2016), the PWIM (Figure 3) addresses word correspondence across sentences, irrespective of their positions. It evaluates pairwise word similarity using three metrics:

cosine distance, L2 Euclidean distance, and dot product, generating a similarity cube where each metric represents a channel. This cube is then processed through a deep convolutional neural network as a feature extractor. While the original paper employs a specific DNN architecture shown below, we have adapted a ResNet version with equivalent depth to leverage residual connections. We hypothesize that, without these connections, the combined depth of the DNN and the underlying BERT model would exacerbate vanishing gradient issues, where our experimental results later begin to substantiate.

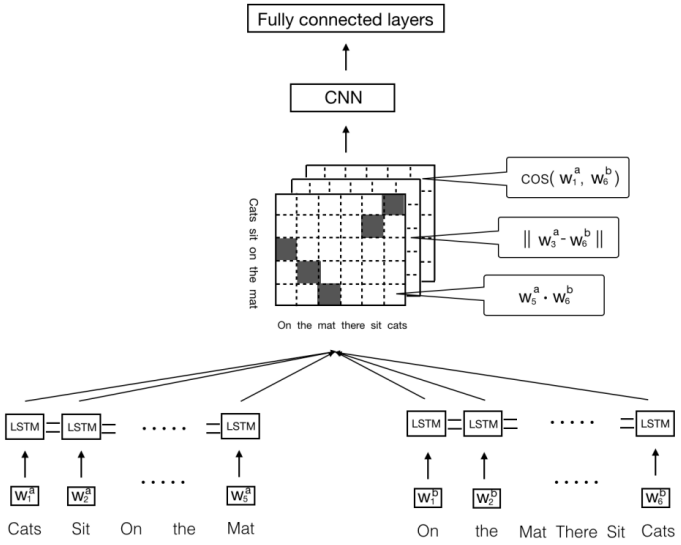

Figure 3: PWIM architecture employing a deep CNN for feature extraction.

2.4 Efficient Fine-Tuning: LoRA and Adapters

Efficient fine-tuning techniques, such as Low-Rank Adaptation (LoRA) and adapters, present innovative approaches to enhance pre-trained models like BERT for specific tasks with minimal computational overhead. These methods introduce additional task-specific layers into the original architecture, allowing for the fine-tuning of a limited subset of model parameters. During the fine-tuning process, the core parameters of the BERT model are frozen, and only the newly added task-specific layers are updated.

Originally devised to curtail the storage burden associated with maintaining separate, fully fine-tuned versions of BERT for each task, the application of LoRA (Hu et al., 2021) and adapter(Houlsby et al., 2019) in this work extends beyond mere efficiency. By focusing fine-tuning efforts on task-specific layers added on top of a high-performing contextual embedding base, these techniques aim to further enhance model performance. The rationale behind this approach is to leverage the refined general capabilities of the pre-trained model while affording the flexibility to tailor its behavior to the nuances of individual tasks through targeted parameter adjustments.

3 Approach

We begin with a baseline model that utilizes the [CLS] token embedding followed by a task-specific linear classifier for each task. BERT model parameters are frozen, with only the classifiers being trainable. Subsequently, we extend this baseline by incorporating several techniques to enhance model performance, as will be detailed in the experimentation section.

3.1 Smoothness Inducing Adversarial Regularisation

The smoothness inducing adversarial regularisation technique is mathematically described as follows:

$$\min_{\theta} F(\theta) = L(\theta) + \lambda_s R_s(\theta), \tag{1}$$

where $L(\theta)$ represents the loss function:

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i; \theta), y_i), \tag{2}$$

with $l(\cdot, \cdot)$ being the task-specific loss function. $\lambda_s > 0$ is a tunable parameter, and $R_s(\theta)$ is the smoothness-inducing adversarial regularizer, defined as:

$$R_s(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} l_s(f(\tilde{x}_i; \theta), f(x_i; \theta)), \tag{3}$$

where $\epsilon > 0$ is another tunable parameter. For classification tasks (sst and quora), the function $f(\cdot; \theta)$ outputs a probability simplex, with l_s chosen as the symmetrized KL-divergence.

$$l_s(P, Q) = D_{KL}(P \| Q) + D_{KL}(Q \| P). \tag{4}$$

For the regression task (sts), $f(\cdot; \theta)$ outputs a scalar within [0,5], with l_s being the squared loss.

As we only implement smoothness inducing adversarial regularisation, so the implementation algorithm is slightly different from the pseudo-code of original paper which exactly implements both parts of SMART algorithm. The pseudo-code for our SMART layer implementation can be referenced to Appendix A

3.2 1D Convolution and Bi-LSTM Layer for SST

For SST, we employ a tailored 1D convolution layer, foregoing the max-pooling layer of the original design, followed by a single-layer Bi-LSTM to capture the intricate word interplay. Each embedding feature is treated as a channel, producing a tensor of the same shape as the input. The output from the 1D convolutional layer is then processed through the Bi-LSTM, with max pooling applied to generate a final sentence embedding vector matching the embedding size of the BERT layer.

3.3 Pairwise Word Interaction Model (PWIM)

We propose two versions of the PWIM, utilizing either a traditional DNN or a ResNet architecture as the feature extractor. For the ResNet version, we utilize residual connections to potentially alleviate the vanishing gradient problem, given the combined depth of the network and the underlying BERT model. Refer to Appendix B for implementation details.

3.4 Further Pre-training with Efficient Fine-tuning

As discussed earlier in this work, we introduce a BERT-based architecture augmented with task-specific layers, alongside a nuanced training paradigm tailored to optimally harness task similarities while accommodating the variance in dataset scales among different tasks. Our approach commences with an initial further pre-training phase employing simplified SMART regularization. This phase targets a BERT-based model supplemented with a Conv1D & Bi-LSTM layer for SST and a shared PWIM layer for both the Quora and STS tasks.

Subsequently, we implement an efficient fine-tuning process wherein only the task-specific components—comprising LoRA, adapters, and task-specific headers are made trainable. This training strategy proceeds in a sequential manner, with tasks being fine-tuned in succession. It is noteworthy that due to the inherent similarity shared by Quora and STS and dataset size disparity between the Quora and STS tasks—with Quora's dataset vastly outnumbering that of STS, the PWIM layer, shared by both tasks, undergoes fine-tuning exclusively with the Quora dataset. Upon transitioning to the STS task, the PWIM layer is then frozen. This approach not only ensures consistency when fine tuning in a sequential manner but also leverages the natural task affinity, thereby fostering a more robust and adaptable model capable of addressing the nuanced requirements of each specific task.

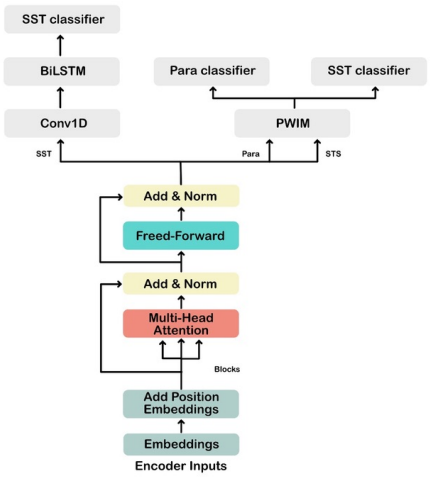

Figure 4: BERT-based model architecture

4 Experiments

4.1 Data

Our foundational minBERT architecture is initially pretrained through two unsupervised learning tasks on Wikipedia content: masked language modeling (MLM) and next sentence prediction. The pretrained model is then used for specialized fine-tuning on three downstream tasks utilizing datasets from the Stanford Sentiment Treebank (SST), Quora, and the SemEval STS Benchmark (STS). The SST dataset has 11,855 single-sentence movie reviews, each categorized across a spectrum from negative to positive sentiments. The Quora dataset contains 400,000 question pairs, each annotated with a binary indicator signifying whether one question rephrases the other. The STS dataset presents 8,628 sentence pairs, each assessed on a scale from 0 (indicating no relation) to 5 (signifying identical meanings).

4.2 Evaluation method

For assessing performance on the SST and Quora tasks, we employ accuracy as the primary metric, reflecting the model's ability to correctly classify each instance within the respective categories.

For the STS task, we utilize the Pearson correlation coefficient to gauge the alignment between the model's predictions and the actual labels, acknowledging the continuous and regression nature of this task.

4.3 Experimentation Details

We detail the experimental setup and configurations employed to evaluate our models:

1. **Early Stopping:** We employed an early stopping strategy for training, applicable in both the baseline model phase and during further pre-training and efficient fine-tuning. The early stopping criteria were set with an improvement threshold of 1×10^{-3} for baseline

and efficient fine-tuning. For the SST and STS tasks, a patience parameter of 5 epochs was utilized, whereas, for the Quora task, the patience was set to 3 epochs. For further pre-training phases, aiming to achieve a highly performant yet generalizable contextual embedding, we adopted an improvement threshold of 1×10^{-2} and a patience of 1 epoch, saving the model only when performance improvements on the validation dataset exceeded the threshold.

2. **SMART Implementation:** The hyperparameters for our SMART implementation adhere to those outlined in the original paper, with the SMART weight λ treated as a tunable parameter within the set $\{3, 5\}$. To determine the optimal perturbation for maximizing the SMART loss, we set the perturbation size (ϵ) to 1×10^{-5}, the noise initialization variance (σ) to 1×10^{-5}, the learning rate for perturbation updates (η) to 1×10^{-3}, and the number of steps for solving the optimization problem to 1.
3. **MT-DNN Configuration:** Following Liu et al. (2019), we implemented a multi-task data loader that, at each step, loads a batch from one of the tasks. The loss weights for each task were inversely proportional to their dataset sizes to address the imbalance across the three downstream tasks.
4. **Handling Different Loss Scales:** The SST task utilized cross-entropy loss, the Quora task employed BCE loss, and the STS task adopted MSE loss. To manage potential scale differences among these losses, we introduced trainable loss weights as a novel solution.
5. **Efficient Fine-Tuning:** In line with the original paper's guidelines, we incorporated LoRA exclusively for the query and value computations, setting the LoRA rank to 4. Adapters were added following both the attention and output layers, with an adapter size of 64.

4.4 Results

Table 1: Performance Metrics Across Different Models

Model Type	SST Acc	Para Acc	STS Corr	Score
Baseline	0.381	0.651	0.187	0.542
MT-DNN+PWIM (dcnn)	0.520	0.742	0.829	0.725
MT-DNN+PWIM (dcnn)+conv1d_bilstm	0.497	0.717	0.813	0.707
MT-DNN+PWIM (resnet)	0.528	0.812	0.863	0.757
MT-DNN+PWIM (resnet)+smart (3)	0.517	0.859	0.846	0.766
MT-DNN+PWIM (resnet)+conv1d_bilstm+smart (3)	0.525	0.848	0.841	0.765
MT-DNN+PWIM (resnet)+conv1d_bilstm+smart (3) + learned_loss_w	0.524	0.847	0.842	0.764
MT-DNN+PWIM (resnet)+conv1d_bilstm+smart (5)	0.529	0.844	0.851	0.766
MT-DNN+PWIM (resnet)+conv1d_bilstm+smart (5)+effi_ft (dev)	**0.531**	**0.861**	**0.870**	**0.776**
MT-DNN+PWIM (resnet)+conv1d_bilstm+smart (5)+effi_ft (test)	**0.529**	**0.871**	**0.858**	**0.776**

For the quantitative analysis, the initial results from the baseline model revealed a huge disparity in performance between the similar tasks of paraphrase detection (Quora) and semantic textual similarity (STS). This highlighted the challenge posed by the small size of the STS dataset compared to the Quora dataset, leading to overfitting in STS and underscoring the importance of leveraging task similarities for improved outcomes.

Transitioning to the Multi-Task Deep Neural Network (MT-DNN) framework and integrating the Pairwise Word Interaction Model (PWIM) for sentence pair tasks significantly enhanced performance across all tasks, where SST even surges a lot from 0.381 to 0.520 because MT-DNN itself is actually a method of generalisation, Quora acc also increases from 0.651 to 0.742 by leveraging the complex PWIM interaction layer and its large dataset for generalisation. The most notbale increase is for STS, which saw a remarkable increase from 0.187 to 0.829. This improvement was largely attributed to the knowledge transfer from the Quora task.

Experimentation with a more complex SST head incorporating a 1D convolution and Bi-LSTM layer to capture the contextually significant interplay, however, resulted in a performance decline due to overfitting, given the limited size of the SST dataset. Additionally, as we employ a loss-weighting strategy based on inverse dataset sizes, it suggested that inefficient loss backpropagation from SST to the underlying BERT model also adversely affected performance of Quora and STS.

The implementation of a ResNet architecture within PWIM, utilizing residual connections, further improved performance, validating our hypothesis on the benefits of enhanced information flow and stable gradient updates with Quora raising from 0.742 to 0.812, and STS from 0.829 to 0.863.

The simplified version of SMART regularization technique then applied during further pre-training yielded even better generalization, with the overall score increased from 0.757 to 0.766, especially Quora increased from 0.812 to 0.859. We then try to tune the SMART weight hyperparameter and add the learned loss weight, but not achieve too much improvement further.

Final fine-tuning stages, incorporating LoRA and adapters and freezing BERT's underlying layers, fine-tuning sequentially for each task in a paradigm we stated in the Approach section, confirmed our approach's efficacy. On the validation dataset, this strategy achieved accuracies of 0.531 for SST, 0.861 for paraphrase detection, and 0.870 for STS correlation, culminating in an overall score of 0.776. Test dataset results mirrored these achievements, with SST Acc = 0.529, Para Acc = 0.871, STS corr = 0.858, and Overall Score = 0.77, underscoring the robustness and effectiveness of our fine-tuning paradigm.

5 Analysis

The majority of our work was to train not only a high-performing (high validation accuracy) model during the further pre-training phase but also to ensure that the contextual embeddings were generalizable. A model displaying high validation performance but near-perfect training accuracy risks forfeiting the potential benefits offered by task-specific layers for further performance enhancements. Thus, we posit that a high-quality, generalizable contextual embedding is one that achieves a given level of validation performance in the fewest possible epochs, with minimal discrepancy between training and validation scores.

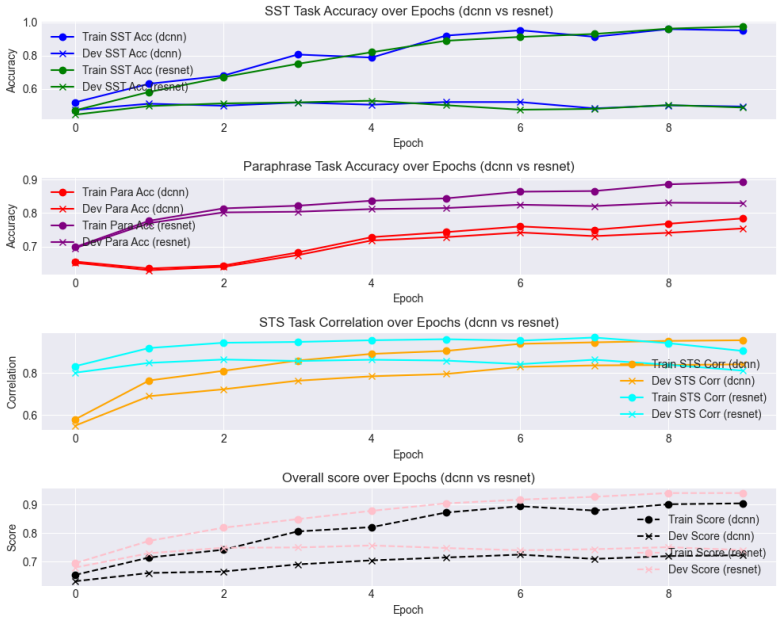

Figure 5: DCNN vs ResNet

Upon examining the impact of employing ResNet and SMART regularization during further pre-training, several observations were made. Firstly, comparing the performance of models using DCNN and ResNet architectures for the paraphrase (para) task revealed a clear advantage for ResNet (Figure 5), showcasing superior performance. In the STS task, while both architectures ultimately delivered comparable results, ResNet achieved this performance in significantly fewer epochs, thereby reducing the risk of overfitting.

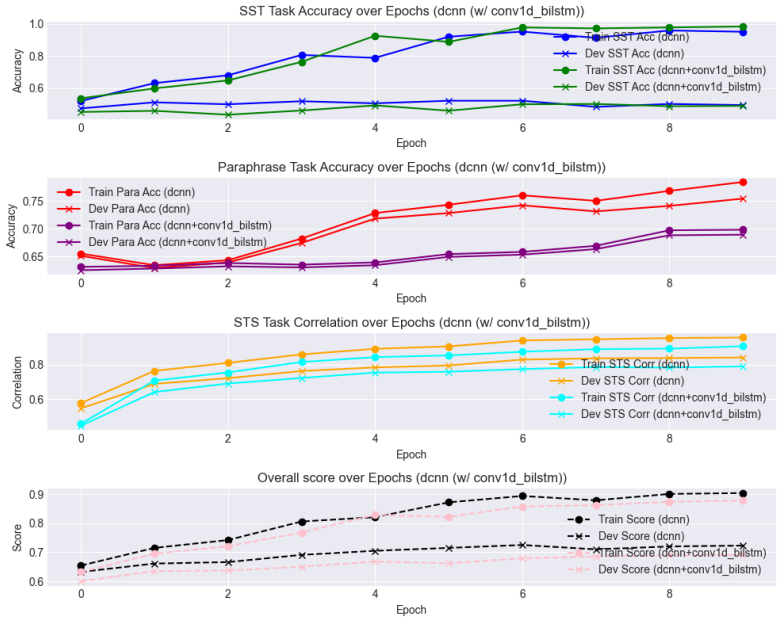

Figure 6: DCNN (w/ conv1d_bilstm)

From figure 6, upon substituting the SST head with a conv1d-biLSTM structure without incorporating SMART regularization, a pronounced increase in training accuracy for SST task was noted, approaching near-perfection. However, this did not translate to improvements in validation performance, suggesting a depletion of potential for further enhancements. This phenomenon of extreme overfitting within the SST task also detrimentally impacted the Quora and STS tasks due to their shared underlying BERT layer.

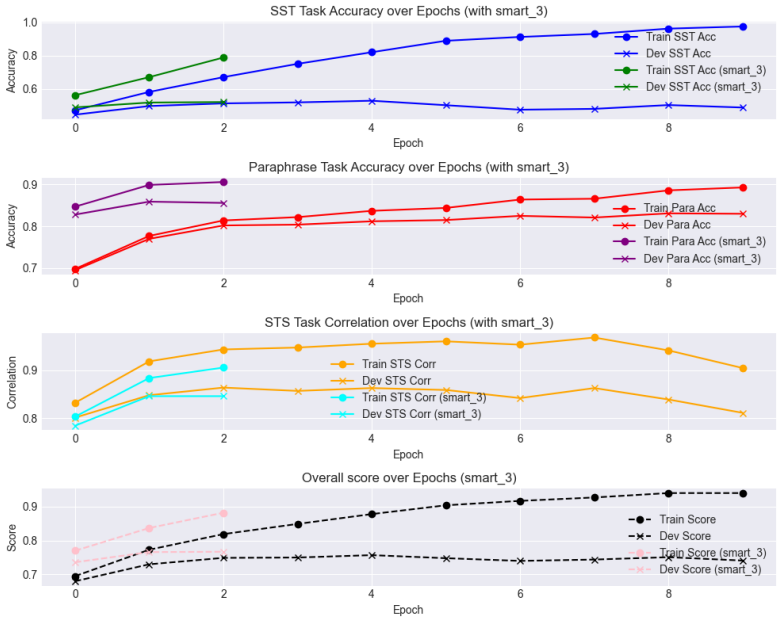

Figure 7: Smart

From figure 7, the application of SMART regularization demonstrated its efficacy: the SST task reached its performance peak within just 3 epochs, compared to 5 without SMART. For the Quora

task, performance goals were met in 3 epochs with SMART, as opposed to 10 epochs without it. Although a slight decline in STS validation performance was observed when employing SMART, the training-validation discrepancy was notably smaller, indicating enhanced generalization. Ultimately, the most robust overall performance, facilitated by SMART, was attained within three epochs—a significant improvement over the 5 epochs required in its absence. This analysis underscores the critical role of advanced architectures like ResNet and regularization techniques such as SMART in crafting a highly generalizable and efficient training paradigm.

6 Conclusion

In our work, we implemented a BERT-based architecture and develop a new paradigm by leveraging the common features in task nature shared by Quora and STS. In order to train a best performing model, when we do further pre-training, we should train a both high-quality and generlisable contextual embedding to avoid depletion of future improvement from efficient fine-tuning at this step. Our future work can attempt very deep CNN architecture in PWIM layer and multi-layer bi-LSTM in SST head to increase the performance further.

References

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, San Diego, California. Association for Computational Linguistics.

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. *arXiv preprint arXiv:2106.09685*.

Hao Jiang, Peng 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. *arXiv preprint arXiv:1911.03437*.

Sakirin Tam, Rachid Ben Said, and Ö. Özgür Tanriöver. 2021. A convbilstm deep learning model-based approach for twitter sentiment classification. *IEEE Access*, 9:41283–41293.

A Appendix

```
def forward(self, batch: dict, state: Tensor, task: str) -> Tensor:
    # Initialization and noise generation
    # for sentence pair tasks (Quora and STS),
    # we need to initialise noise for each input sentence embedding
    noise = torch.randn(batch_size, seq_len, embed_size) * noise_var

    for i in count():
        # Compute perturbed states
        state_perturbed = eval_fn[task](token_ids, attention_mask, noise=noise)
        if last step:
            return final loss
        # Compute perturbation loss and update noise
        loss = loss_fn[task](state_perturbed, state)
        noise_gradient, = torch.autograd.grad(loss, noise)
        # Move noise towards gradient to change state as much as possible
        step = noise + step_size * noise_gradient
        # Normalize new noise step into norm induced ball
```

```
step_norm = norm_fn(step)
noise = (step / step_norm) * epsilon
# Reset noise gradients for next step
noise = noise.detach().requires_grad_()
```

B Appendix

Layer Type	Specifications
Convolutional Layer 1	`nn.Conv2d(3, 128, kernel_size=3, stride=1, padding=1)`
Batch Normalization 1	`nn.BatchNorm2d(128)`
ReLU Activation 1	`F.relu()`
Max Pooling 1	`nn.MaxPool2d(kernel_size=2, stride=2)`
Dropout 1	`nn.Dropout(0.5)`
Convolutional Layer 2	`nn.Conv2d(128, 164, kernel_size=3, stride=1, padding=1)`
Batch Normalization 2	`nn.BatchNorm2d(164)`
ReLU Activation 1	`F.relu()`
Max Pooling 1	`nn.MaxPool2d(kernel_size=2, stride=2)`
Dropout 2	`nn.Dropout(0.5)`
Convolutional Layer 3	`nn.Conv2d(164, 192, kernel_size=3, stride=1, padding=1)`
Batch Normalization 3	`nn.BatchNorm2d(192)`
ReLU Activation 1	`F.relu()`
Max Pooling 1	`nn.MaxPool2d(kernel_size=2, stride=2)`
Dropout 3	`nn.Dropout(0.5)`
Convolutional Layer 4	`nn.Conv2d(192, 192, kernel_size=3, stride=1, padding=1)`
Batch Normalization 4	`nn.BatchNorm2d(192)`
ReLU Activation 1	`F.relu()`
Max Pooling 1	`nn.MaxPool2d(kernel_size=2, stride=2)`
Dropout 4	`nn.Dropout(0.5)`
Convolutional Layer 5	`nn.Conv2d(192, 128, kernel_size=3, stride=1, padding=1)`
Batch Normalization 5	`nn.BatchNorm2d(128)`
ReLU Activation 1	`F.relu()`
Adaptive Max Pooling	`nn.AdaptiveMaxPool2d((1, 1))`
Dropout 5	`nn.Dropout(0.5)`
Fully Connected Layer 1	`nn.Linear(128, 1024)`
Fully Connected Layer 2	`nn.Linear(1024, output_classes)`

Table 2: DNN Architecture for PWIM

Layer Type	**Specifications**
Residual Block 1	Projection (if applicable) Conv: `nn.Conv2d(128, 128, kernel_size=3, stride=1, padding=1)` Batch Norm: `nn.BatchNorm2d(128)` ReLU: `F.relu` Dropout: `nn.Dropout(0.5)` Conv: `nn.Conv2d(128, 128, kernel_size=3, stride=1, padding=1)` Batch Norm: `nn.BatchNorm2d(128)` Dropout: `nn.Dropout(0.5)` Residual Connect ReLU: `F.relu`
Residual Block 2	Same as Block 1
Residual Block 3	Same as Block 1
Residual Block 4	Same as Block 1
Residual Block 5	Same as Block 1
Adaptive Max Pooling	`nn.AdaptiveMaxPool2d((1, 1))`
Fully Connected Layer 1	`nn.Linear(128, 1024)`
Fully Connected Layer 2	`nn.Linear(1024, output_classes)`

Table 3: ResNet Architecture for PWIM