Exploring Pretraining, Finetuning and Regularization for Multitask Learning of minBERT

Stanford CS224N Default Project

Brian Song*, **Xinyu Hu***, **Zhiyin Pan***
Department of Computer Science
Stanford University
{wsong20, huxinyu, zhiyinp}@stanford.edu

Mentor: Timothy Dai, No external collaborators or project sharing

Abstract

Fine-tuning a large, pre-trained model like BERT (Devlin et al., 2019) is the convention for doing downstream natural language understanding tasks. However, effectively adapting a single base model for multiple tasks presents challenges, as the optimal parameters for one task may conflict with those of another, leading to potential performance degradation. In this project, we explored a combination of techniques to enhance model performance and adaptability across tasks. We implemented a multi-task learning baseline with BERT and investigated contrastive sentence embedding (SimCSE), projected attention layers (PALs), and SMART regularization as its extensions. Our findings demonstrate that PALs consistently improved performance, showcasing their ability to learn task-specific representations. While SimCSE did not yield significant gains in this context, SMART regularization successfully provided a performance boost by introducing less aggressive updates. Finally, strategically ensembling models led to further advancements in our multi-task NLU results by improving the generalization. We achieved our best average dev set accuracy of 0.786 using a 7 model ensemble.

1 Introduction

Since Vaswani et al. (2017) introduced the Transformer architecture for natural language processing (NLP), the paradigm of pretraining and subsequently fine-tuning large pre-trained models, such as BERT (Devlin et al., 2019), has established itself as a cornerstone for achieving state-of-the-art (SOTA) results across a multitude of NLP tasks. Notably, benchmarks like the Stanford Sentiment Treebank (Socher et al., 2013) have witnessed remarkable performance enhancements through the task-specific tuning of these large-scale models. Yet, the prospect of employing a singular model across multiple tasks presents a compelling avenue for exploration, primarily due to its potential to maximize a model's generalization capabilities. A critical question arises: Can tuning a single model deliver near-SOTA results across various tasks? This inquiry brings numerous technical challenges, such as hard-sharing model weights leads to overfitting to specific tasks. This project endeavors to navigate the end-to-end process of training a BERT model, aiming to identify and address these challenges.

In the domain of enhancing sentence embeddings, the SimCSE approach introduced by Gao et al. (2021) employs contrastive learning methods during the pretraining phase to boost model performance. The PAL methodology, as elucidated by Stickland and Murray (2019), integrates additional task-specific weights into the original BERT self-attention mechanism, thereby optimizing the model's multitasking finesse during fine-tuning. Furthermore, the SMART regularization technique proposed by Jiang et al. (2020) presents a robust mechanism to counteract the issue of overfitting associated

*These authors contributed equally to this work

with hard-shared weights. By integrating the SimCSE strategy in the pretraining phase, applying PAL during fine-tuning, and implementing SMART throughout the model training process, we posit that these combined interventions will effectively address the technical challenges at hand, thereby facilitating significantly enhanced outcomes.

2 Related Work

SimCSE(Gao et al., 2021)
Learning universal sentence embeddings is a fundamental problem in natural language processing. Contractive learning seeks to develop effective representation by bringing semantically similar neighbors closer and pushing apart non-neighbors. Wang and Isola (2020) highlight two essential properties of contrastive learning—alignment and uniformity—and propose their use in evaluating representation quality. According to their framework, positive instances should remain close, while the embeddings of random instances should be distributed uniformly across the hypersphere.

SMART (Jiang et al., 2020)
Due to the scarcity of data from downstream tasks and the vast capacity of pre-trained models, aggressive fine-tuning frequently results in overfitting and loss of knowledge acquired during pre-training. The SMART paper (Jiang et al., 2020) proposes a new framework aimed at fine-tuning pre-trained language models in a robust and efficient manner. With correct hyperparameter settings, it serves a strong regularizer, preventing overfitting during multitask training.

PALs (Stickland and Murray, 2019)
Stickland and Murray (2019) explored how to adapt a single large base model to work with multiple tasks by adding low-dimensional multi-head attention layers (PALs) in parallel with the standard BERT layers, tailored to each task. This approach enables task-specific global attention without a significant increase in the number of parameters. They also introduced a novel method for scheduling training, where they sample tasks proportional to their training set size at first, and de-emphasize training set size as training proceeds. PALs achieved comparable performance to that of individually fine-tuned BERT models on the GLUE benchmark while utilizing approximately 7x fewer parameters.

Model Soup (Wortsman et al., 2022)
Ensemble methods (Dietterich, 2000) have long been a established paradigm in Machine Learning. Wortsman et al. (2022) applied similar ensemble techniques to a collection of fine-tuned attention-based vision models, achiving state-of-the-art results on ImageNet (Deng et al., 2009) classification tasks. We want to adpot this methodology and see if is also works for language models.

3 Approach

3.1 Main Approach

In the initial phase of our project, we aimed to establish a performance baseline using multi-task classifiers built on a shared BERT model. We added three task-specific classifiers, each comprising of two feed forward layers with GELU (Hendrycks and Gimpel, 2016) activation function. The size of the intermediate layer was set to match the BERT base layer, at 768 units. For the final output layer, we utilized an additional linear layer to produce logits corresponding to the tasks' labels. One thing worth noting is that, we concatenated the embeddings of sentence 1 and sentence 2 for paraphrase detection and semantic textual similarity tasks. We believe that the concatenation can help to let the model refers both sentences during training.

In the second phase, we evaluated the impact of three extensions (PAL, SimCSE, and SMART) on downstream tasks, experimenting with various combinations of these methods. Our exploration included tailored modifications to adapt these extensions more effectively.

For the final stage of this project, we ensembled multiple models together to create a more robust system capable of performing well across all three tasks. Note that each models has been fine-tuned on all three tasks, despite differences in hyperparameters and model structures. We had assigned prior probability weights to different models, treating these weights as hyperparameters to be optimized through experimentation. For model selection, we employed a greedy-based approach, as introduced by Wortsman et al. (2022). This method involves ranking the models by their accuracy for each task

and incorporating them into the ensemble if they contribute to an overall improvement in accuracy. (See the appendix for pseudocode.) For the ensembling methods:

Weighted average	$f = \sum_{i=1}^{k} W_i f(x;\theta_i) / \sum_{i=1}^{k} W_i$
Weighted majority vote	$f = C_j,$ where $\sum_{f(x;\theta_j)=C_j} W_j > \sum_{f(x;\theta_i)=C_i} W_i$ for other class i

3.2 Extensions

SimCSE Unsupervised
We implemented unsupervised SimCSE as described in (Gao et al., 2021), to evaluate the downstream performance of the three classifiers with BERT embeddings vs. SimCSE pre-trained embeddings. The idea of unsupervised SimCSE is extremely simple: we take a collection of sentences and duplicate them. Then we feed these sentences into BERT model to get two embeddings. Due to applying different dropout masks, these two embeddings are similar but not the same; The network is then trained with the following objective function:

$$\ell_i = -\log \frac{e^{sim(h_i,h_i')/\tau}}{\sum_{j=1}^{N} e^{sim(h_i,h_j)/\tau} + \sum_{j=1}^{N} e^{sim(h_i,h_j')/\tau}} \tag{1}$$

We experimented with various datasets, including 1 million wiki sentences as per Tianyu Gao (2022), and also all the sentences in the training data + normalized 1m wiki sentences with the same sentence length (256+ characters).

SMART (Jiang et al., 2020)
The scarcity of training data could result in aggressive updates for our model during fine tuning, leading to overfitting. SMART provides a strong regularization effect on model updates through two main strategies: The first involves Smoothness-Inducing Adversarial Regularization: Introduces small perturbations to, and encourage the output to not change too much. This is aimed at smoothing the output function f across the neighborhoods of all input x_i. Specifically, this is eqivalent to solving the following optimization for fine-tuning:

$$\min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta) \tag{2}$$

, where $\mathcal{L}(\theta)$ is the original loss function, and $\mathcal{R}_s(\theta)$ is the SMART loss term defined as:

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

Here, ℓ_s stands for the symmetrized KL-divergence for classification task, and squared loss for regression task. The second strategy uses Bregman Proximal Point Approximation to solve (2). This method also serves as a strong regularizer to prevent aggressive updates each step when the regularization parameter μ is large. The Vanilla Bregman Proximal Point methods is:

$$\theta_{t+1} = \text{argmin}_\theta \mathcal{F}(\theta) + \mu \mathcal{D}_{\text{Breg}}(\theta, \tilde{\theta}_t) \tag{4}$$

The implementations we have integrated are based on the original implementation [1] and this GitHub repository[2]. (Jiang et al., 2020)

PALs (Stickland and Murray, 2019)
As described in the Related Work section, Stickland and Murray (2019) proposed projected attention layers (PALs) to improve the accuracy of multi-task learning of a single base model. The main idea behind PALs is to add low-dimensional multi-head attention layers in parallel to the standard BERT layers, specific to each task. We implemented this architecture from scratch, but referenced their original code that is available in this[3] GitHub repository.

For original BERT architecture, the input hidden states h to each BERT layer undergo a multi-headed self-attention operation *SA*, a residual connection, and layer normalization *LN* to get h_{att}. This is then passed through a feed-forward network *FFN*, followed by another residual connection and layer normalization *LN* to get the final output of each BERT layer, represented as *BL* by the equationss:

$$h_{\text{att}} = \textit{LN}(h + \textit{SA}(h)), \textit{BL}(h) = \textit{LN}(h_{\text{att}} + \textit{FFN}(h_{\text{att}})) \tag{5}$$

[1] https://github.com/namisan/mt-dnn

[2] https://github.com/archinetai/smart-pytorch

[3] https://github.com/AsaCooperStickland/Bert-n-Pals

PALs add low-dimensional multi-head attention layers in parallel to the standard BERT layers. The input hidden states h undergo a lower dimension projection by an encoder matrix V_E, a multi-headed self-attention operation *SA* and an upper dimension projection by a decoder matrix V_D to obtain the task-specific functions *TS*. *TS* is added in parallel to the standard BERT layers to get the final output, represented as *PAL* by the following equations.

$$TS(h) = V_D(SA(V_E(h))), PAL(h) = LN(h_{\text{att}} + FFN(h_{\text{att}}) + TS(h)) \tag{6}$$

By stacking 12 layers of the modified BERT layer with PAL on top of each other, similar to the original BERT model, the modified BERT model with PALs is completed.

PAL Scheduling Stickland and Murray (2019)
We also implemented the PAL scheduling as introduced by Stickland and Murray (2019). The scheduling approach involves initially sampling tasks proportional to their training set size and gradually de-emphasizing training set size as training progresses. We sample each task i with a probability p_i which is proportional to the task's training set size N raised to the power α. The value of α is introduced so that we can de-emphasize training set size as training proceeds because it is beneficial to train on tasks more equally towards the end of training. The value of α changes with each epoch e and E (the total number of epochs).

$$p_i \propto N^{\alpha}, \alpha = 1 - 0.8\frac{e-1}{E-1} \tag{7}$$

3.3 Original Explorations

SimCSE Unsupervised Gao et al. (2021)
We want to evaluate the impact of sentence length for SimCSE (Gao et al., 2021), considering the same sentence always has the same length and BERT encodes sentence length. In addition, we are curious about how different training data would have impacted downstream performance. We trained and evaluated the following variances: 1) 1 million wiki sentences Tianyu Gao (2022); 2) a subset of the 1 million wiki dataset that are longer than 256 and normalized those sentences to have a length 256; 3) all training sentences from these datasets: a) Stanford Sentiment Treebank (SST) (Socher et al., 2013). b) Quora dataset. c) SemEval STS Benchmark dataset (Agirre et al., 2013).

Frozen BERT with PALs Training (Stickland and Murray, 2019)
One limitation of PALs Training is that it only concentrated on training all the parameters instead of freezing the base BERT parameters and only fine-tuning the PALs' parameters. In our project, we explored this as one of our originalities. We also observe some potential weight interference during fine-tuning between different task training with the original PALs. Consequently, we conducted this experiment, which froze the BERT weights and exclusively trained task-specific PALs and classifiers to mitigate such interference. We documented the experiment results in the next section, and it is a valuable component of our ensemble model as it helps mitigate weight interference between tasks.

4 Experiments

4.1 Data

We trained our models on the provided datasets from the default final project handout: 1) Stanford Sentiment Treebank (Socher et al., 2013) dataset for training sentiment analysis task (SST). 2) The Quora dataset for training paraphrase detection task (Para). 3) SemEval STS Benchmark dataset for training semantic textual similarity task (STS). 4) 1 million sentences from Wiki (Tianyu Gao, 2022), which is the SimCSE training dataset included by the original paper. (Gao et al., 2021)

4.2 Evaluation method

We use the accuracy on the SST dev/test dataset, the accuracy on the Quora dev/test dataset, and the Pearson score on the SemEval STS Benchmark dev/test dataset, as well as an aggregate score across all three tasks to evaluate how well our model is performing.

4.3 Experimental details

Base Model Tuning We investigated the impact of the number of hidden layers and the size of hidden states in classifiers on model performance. Our findings indicated a general decline in performance compared to the baseline. This suggests that simply increasing model complexity does not necessarily enhance performance and may, in fact, lead to overfitting.
In examining various classifier architectures, we discovered that for paraphrase detection and semantic textual similarity tasks, concatenating the embeddings of sentences 1 and 2 proved most effective. We believe this is because embeddings concatenation allows the model to reference both sentences during training, facilitating a better understanding of their relationship.
We observed that the dataset for paraphrase detection was disproportionately larger than those for the other two tasks. To address this, we experimented with adding an SST training run in end of each epoch. This adjustment led to improved model performance, suggesting that imbalanced training data among tasks sharing weights can negatively affect overall performance, as measured by our specific evaluation method.
We further tuned hyperparameters including dropout rate and weight decay, etc. It resulted in some model performance improvements.

Model	SST	Paraphrase	STS	Dev Score
Fine tune Initial base model	0.425	0.742	**0.861**	0.699
Fine tune*	**0.499**	**0.862**	0.860	**0.764**

Table 1: Baseline Performance Tuning without Extensions

SimCSE We implemented unsupervised SimCSE described in (Gao et al., 2021), and explore the following datasets: 1) 1 million wiki sentences Tianyu Gao (2022); **training time**: 6 hours per epoch. 2) a subset of 1m wiki dataset that are longer than 256 characters, and normalized those sentences to be 256; **training time**: 2 hours per epoch, 3) 303,617 sentences extracted from training sets: a) Stanford Sentiment Treebank (SST) (Socher et al., 2013); b) Quora dataset; c) SemEval STS Benchmark dataset Agirre et al. (2013); **training time**: 2 hours per epoch. We explored **some hyperparameter tuning** including: 1) attention dropout probability (0.1 and 0.3) 2) pooling methods (last hidden state and Cls), with **learning rate** 1e-5 and **weight decay** 0.01, 0.05, 0.1.

SMART The hyperparameters we used for the SMART framework are: $S = 1$, $T_{\tilde{x}} = 1$, perturbation norm parameter $\epsilon = 10^{-6}$, perturbation standard deviation $\sigma = 10^{-6}/10^{-5}$, $\mu = 0.02/2$, and $\lambda_s = 1$. The optimizer we use is AdamW, with $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Overall optimizer learning rate is 10^{-5}. We have been running fine-tuning with SMART implementation on top of PALs and SimCSE pre-trained embeddings for 10 epochs. Each epoch takes **1.5h** to complete.

PALs The hyperparameters for PALs are what recommended by Stickland and Murray (2019): the size of each PALs attention layer is $d_s = 204$, with 12 attention heads per PAL layer.

Ensembling We have pre-trained and fine-tuned various models across all three tasks, employing a greedy-based strategy for model ensembling. We have done 2-model, 5-model and 7-model ensemble for dev & test set evaluation.

4.4 Results

SimCSE
We expected SimCSE to help improve the downstream task results. However, adding SimCSE results in a 0.001 accuracy drop against the baseline, which shows that adding SimCSE embeddings has no obvious impact. We expected training SimCSE with normalized sentence length to improve embeddings distribution and thus help the downstream tasks' performance. Our experiments resulted in a 0.035 accuracy drop against the baseline, showing normalized sentence length SimCSE does not improve our accuracy.
We were curious about the impact of training SimCSE with adapted data from SST, STS, Paraphrase datasets. We were worried about overfitting, as the test dataset sentences won't have the same

*Performance gain comes from a combination of 1) adding one more SST training per epoch; 2) changing paraphrase model architecture; 3) hyper parameters tuning (dropout, weight decay).

pre-processing. Our experiments show that indeed overfitting happened, as the downstream classifier performance went down.

Model	SST	Paraphrase	STS	Dev Score
Pretrain base model	0.454	0.625	0.546	0.617
Finetune base model	0.499	0.862	0.860	0.764
+ SimCSE	0.501	0.852	0.870	0.763
+ SimCSE normalized sentence length	0.519	0.745	0.849	0.729
+ PALs	0.513	0.862	**0.871**	0.770
+ PALs + PAL scheduling	0.517	0.860	0.862	0.769
+ Frozen BERT with PALs training	0.506	0.625	0.797	0.676
+ SMART (weak)	0.510	0.861	0.864	0.768
Combination Models				
+ SimCSE + PALs	0.526	0.867	0.867	0.775
+ SimCSE + PALs + SMART (weak)	0.505	0.863	0.847	0.764
Best Single Model				
+ SimCSE + PALs + SMART (strong)	**0.531**	**0.869**	0.864	**0.778**

Table 2: Extension Performance Comparisons

PALs
PALs enhanced our base model's performance across tasks, yielding a 0.006 accuracy increase as expected. The improvement comes from PALs storing task-specific weights, improving robustness. Although we expected PALs with PAL scheduling to further enhance the paraphrase detection task considering its very large dataset, it unexpectedly led to a 0.001 accuracy decrease compared to the base PALs model. This discrepancy may stem from potential imbalances within the paraphrase detection dataset, despite its size, affecting the effectiveness of PAL scheduling.
Regarding frozen BERT weights with PALs training, we anticipated it to improve upon the pre-trained version but not surpass the fine-tuned variation. Our experiments confirmed this expectation, with a performance of 0.676 compared to 0.617 and 0.764 for the base pre-trained and fine-tuned models, respectively. While PALs during pre-training can enhance multi-task capabilities, fine-tuning allows all of BERT's layers to adapt to target tasks' nuances, resulting in superior performance.

SMART
We expected SMART to provide good regularization results and boost performance. Adding SMART with weak regularization settings showed a near-neutral 0.002 performance boost from the base fine-tuned model while a strong regularization combined with other techniques produced our best single model (+0.014 from the baseline). Our findings align with our expectations.

Ensembling results
We have run 2-model, 5-model, and 7-model ensemble on many variation of our base models trained on all three tasks. We could observe from the above table that the model performance improves gradually as we incorporates more models into the ensemble for the weighted average method. Comparing table 3 to table 2, ensembling boosts our overall performance on all three tasks. Our best overall performance among ensembled models (and also single models) are achieved by a 7-class ensemble with overall accuracy 0.785.

Ensemble Model Spec	SST	Paraphrase	STS	Dev Score
2-model Weighted Average	0.533	0.873	0.872	0.781
2-model Majority Vote	0.507	0.870	0.872	0.771
5-model Weighted Average	0.537	0.872	0.879	0.783
5-model Majority Vote	0.505	0.497	0.879	0.647
7-model Weighted Average	**0.542**	**0.876**	**0.882**	**0.786**
7-model Majority Vote	0.538	0.869	0.882	0.783

Table 3: Model ensembling Performance Comparisons
Observe that for paraphrase and SST, using majority vote approch actually shows significant performance drop, despite the same config for ensembling. The 5-class majority vote method results

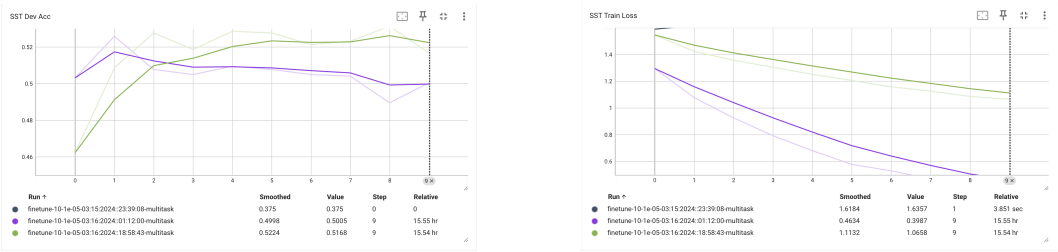
in a drastic 0.373 accuracy drop in paraphrase detection. This might indicate that by incorporating majority vote we are actually forcing a hard boundary for the prediction result and might not provide a good estimation on the real distribution.

5 Analysis

SimCSE: Observed no **Positive Impact**
1) Our experiments showed no obvious downstream task performance impact using SimCSE pre-trained weights on 1 million wiki sentences for 3 epochs. We think training for more epochs might help improving sentence embeddings. 2) We expected training SimCSE with normalized sentence length would improve our embeddings distribution and thus boost the downstream classifier performance. However, our experiments did not support our speculation. We infer that our particular method might be to blame, as limiting the sentences to be longer than 256 characters reduced the training data to 1/3. Further analysis would be necessary.
Negative Impact We were curious about the impact of training SimCSE with adapted data from SST, STS, and Paraphrase datasets. We are concerned with overfitting, as the test dataset sentences won't have the same pre-processing. Our experiments show degraded performance on downstream tasks, indicating potential overfitting.

PALs: Our experiments revealed several key insights into the strengths and limitations of PALs within our multi-task learning system. Firstly, the addition of PALs consistently improved performance across tasks compared to the baseline model. This suggests that PALs effectively capture task-specific knowledge and attention patterns. However, our anticipation that PALs with PAL scheduling would further refine the paraphrase detection task's accuracy was contradicted by our findings. Despite the plentiful training data available for this task, the efficacy of PAL scheduling seemed hindered by potential imbalances within the dataset, suggesting that variations in the quality or diversity of paraphrases may have impeded its effectiveness. Lastly, while frozen BERT with PALs showed improvement over the pre-trained baseline, it predictably underperformed compared to the fine-tuned model. This reinforces the importance of full model adaptability for optimal task-specific performance, even though PALs can enhance the pretrained model's multitasking capabilities.

SMART analysis: Our results shows SMART regularization framework is providing model performance boost when we set a reasonable hyperparameter. We think it boosts our model performance by introducing less aggressive updates. Previously we had a problem that sentiment analysis task converging to the best result in 2-3 epochs, and often observes a performance drop when the epoch goes up. However, when we set a reasonable parameter and enable the SMART framework, we actually observed less aggressive updates to the model weight and the sentiment classification dev accuracy was able to converge later and achieve a higher result.



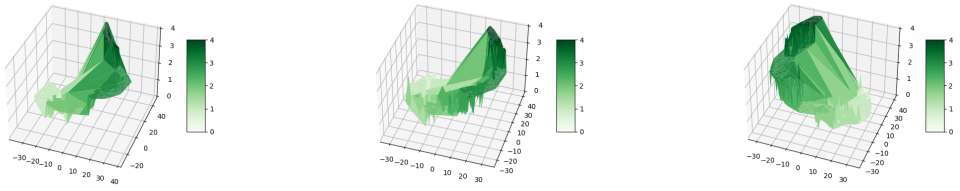
(a) Sentiment Analysis Dev Accuracy (b) Sentiment Analysis Dev Loss

Figure 1: Sentiment Analysis dev result with multi-task SMART setting, purple represents $\epsilon = 10^{-6}, \lambda_s * \mu = 0.2$ and green represents $\epsilon = 10^{-5}, \lambda_s * \mu = 2$.

From the above graph, we could see that sentiment analysis accuracy still peaks at epoch 2-3 for a weaker SMART setting. However when we increase the SMART regularization strength, we actually see train loss decrease much slower, while dev accuracy gradually increases, peaking at epoch 9-10. This shows SMART is indeed providing a strong regularization effect with the correct settings. This matches the result finding from the paper: "We only observed slight differences in model performance when $\lambda_s \in [1, 10], \mu \in [1, 10]$ and $\epsilon \in [10^{-5}, 10^{-4}]$; when $\lambda_s \leq 0.1, \mu \leq 0.1$ or

$\epsilon \leq 10^6$, the regularization is unreasonably weak." (Jiang et al., 2020)

Ensembling Analysis with PCA and t-SNE Incorporating multiple base models significantly enhances our model's performance, and we aim to empirically understand the reasons behind this improvement. To achieve this, we extract the BERT output embeddings and employ PCA and t-SNE for unsupervised dimensionality reduction, reducing the embedding dimension from 768 to 2. These reduced dimensions serve as the base 2-D coordinates. Subsequently, we depict the sentiment prediction results by varying the height. This visualization is performed for our single model, as well as for our 2-model and 5-model ensembles, yielding the results presented below:



(a) Single model (b) 2-model ensemble (c) 5-model ensemble

Figure 2: Sentiment Analysis result plot with PCA and t-SNE

The above figures suggest that ensembles comprised of five models occupy a larger volume in the embedded space compared to ensembles with one or two models. This observation aligns with the notion that while individual models may excel in capturing specific regions of the prediction landscape, ensembles effectively integrate diverse predictive areas. Consequently, the resulting ensemble model exhibits a more robust performance.

6 Conclusion

Achievements We successfully implemented a multi-task learning baseline with BERT. We explored and evaluated the effectiveness of SimCSE, PALs, and SMART regularization for multi-task learning. We demonstrated the benefits of PALs and SMART regularization in improving multi-task performance.

Key Findings
1) Increased model complexity alone may not always guarantee better performance. Our initial exploration with larger hidden layers in classifiers resulted in a performance decline and led to potential overfitting.
2) SimCSE did not significantly improve downstream tasks. While we explored different training datasets and hyperparameters, SimCSE did not lead to consistent improvements.
3) PALs effectively captured task-specific knowledge and improved performance. Models with PALs consistently outperformed the baseline across all tasks. PAL scheduling did not show significant benefits for paraphrase detection. The potential imbalances within the paraphrase dataset might have hindered its effectiveness. Frozen BERT with PALs training offered some improvement but it underperformed compared to fine-tuning. This reinforces the importance of full model adaptability for optimal performance.
4) SMART regularization provided a performance boost. By introducing less aggressive updates, SMART helped the sentiment analysis task achieve a higher accuracy later in training.
5) Ensemble effectively integrate the predictions/classifications from various models, resulting in a stronger model and a more reliable outcome.

Limitations and Future Work 1) Explore alternative methods, such as supervised SimCSE, for pre-training sentence embeddings, potentially with larger datasets or longer training times. 2) Experiment with different combinations of SimCSE, PALs, and SMART, and explore their impact on a wider range of multi-tasking scenarios. 3) The effectiveness of PAL scheduling requires further investigation, particularly regarding dataset quality and diversity. Further investigate PAL scheduling, particularly regarding dataset quality and diversity. 4) Further investigate interference reduction from training on separate tasks. Gradient surgery can be a natural extension to further improvement.

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.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs).

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

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.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Danqi Chen Tianyu Gao, Xingcheng Yao. 2022. SimCSE: Simple Contrastive Learning of Sentence Embeddings.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.

7 Appendix (optional)

Algorithm 1 Greedy model soup

procedure GREEDY MODEL SOUP($\theta_1, \ldots, \theta_k$)
    Potential soup ingredients $Cand = \{\theta_1, \ldots, \theta_k\}$
    Soup prior probability $Weight = \{w_1, \ldots, w_k\}$
    Final soup ingredients $Fin = \{\}$
    for $i = 1$ to k **do**
        if Accuracy(average($Fin \cup w_i\theta_i$)) > Accuracy(average(Fin)) **then**
            $Fin = Fin \cup \theta_i$
        end if
    end for
    return Fin.
end procedure

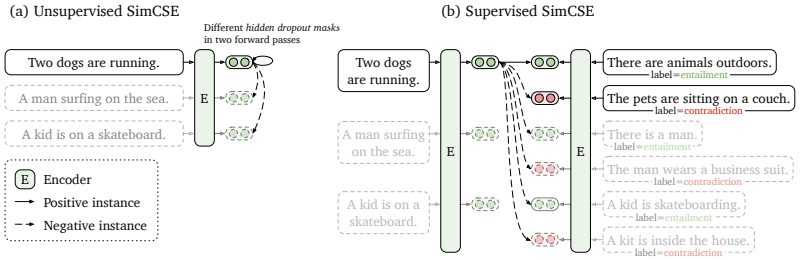


Figure 3: SimCSE Model structureGao et al. (2021)