Extending BERT with Multi-task and Meta-learning

Stanford CS224N Default Project; Mentor: Anuj Nagpal , No External Collaborators, No Sharing project

Jing Ning
Department of Computer Science
Stanford University
annaning@stanford.edu

Cam Burton
Department of Computer Science
Stanford University
cam21@stanford.edu

Abstract

Multi-task learning is a machine learning approach that trains a model to perform multiple related tasks simultaneously, instead of training separate models for each task. Our project involved the implementation of a range of techniques aimed at improving model performance, which resulted in a 21.9% increase in accuracy score in the development set and 20.4% in the test set. The methods utilized included a combination of: 1) Using projected attention layers as shared weights to improve performance 2) A new improved Siamese network architecture designed to improve information flow for cosine similarity tasks and paraphrase tasks 3) hyperparameter tuning that involved the adjustment of multi-task loss weights, learning rate scheduler, and loss function, and 4) fine-tuning the model using an additional dataset to address issues of overfitting and 5)We also employ an ensemble approach by combining the best-performing models for each dataset and observe that the ensemble technique leads to additional improvements in performance.

Furthermore, we introduce a new Proto BERT meta-learning network architecture that is customized for the sentiment analysis task, achieving a promising result of 40.4% accuracy on the 5-shot meta-learning test.

1 Introduction

Recent advances in natural language processing (NLP) have led to significant improvements in various tasks such as sentiment analysis, question answering, and machine translation. One of the most successful models in this field is BERT (Bidirectional Encoder Representations from Transformers), which leverages a transformer-based architecture to learn contextualized representations of words.BERT is pretrained on a corpus and this allows BERT to understand the nuances of language and develop a deep understanding of context and meaning, making it well-suited for tasks such as sentiment analysis. However, while BERT has shown impressive results on a wide range of tasks, it still suffers from some limitations. Specifically, BERT was originally designed for a single task and may not generalize well to new tasks or domains. To address this issue, we propose to extend BERT with two techniques: multitask learning and meta-learning. By training BERT on multiple related tasks simultaneously, we aim to improve its ability to generalize to new tasks. Additionally, by using meta-learning, we hope to improve the efficiency and effectiveness of the training process, enabling us to learn from fewer examples and adapt more quickly to new tasks.

2 Related Work

BERT (Bidirectional Encoder Representations from Transformers) [1] is a pre-trained deep learning model for natural language processing (NLP) that was introduced by Devlin et al. in 2018 . BERT is based on the Transformer architecture proposed by Vaswani et al.[2] in 2017 , which has revolutionized the field of NLP by allowing for more efficient and effective processing of sequences of data.

Since its introduction, BERT has been widely adopted and has achieved state-of-the-art performance on a wide range of NLP tasks, such as question answering, sentiment analysis, and language translation . BERT has also inspired many subsequent research works that aim to improve upon or extend its capabilities.

One of the key areas of research related to BERT has been multi-task learning, which involves training a single model to perform multiple related tasks simultaneously. This approach has been shown to improve model performance and reduce the amount of training data required . Several works have explored multi-task learning with BERT, such as Liu et al [3] who proposed a method for jointly training BERT on multiple tasks using a shared encoder and task-specific classifiers.

Another area of research related to BERT has been meta-learning, which involves training a model to learn how to learn. This approach has been shown to be particularly effective in few-shot learning scenarios where only a small amount of data is available for training. Several works have explored meta-learning with BERT, such as Lee et al [4] who proposed a meta-learning approach for fine-tuning BERT on low-resource tasks.

More recently, there have been several works that aim to extend BERT's capabilities beyond language modeling. For example, Zhang et al. [5] a method for using BERT to perform cross-modal retrieval tasks, such as matching images with text descriptions. Li et al. [6] proposed a method for using BERT to perform entity linking, which involves identifying mentions of entities in text and linking them to a knowledge base.

3 Approach

3.1 Baseline

BERT [1] is pretrained on a corpus and this allows BERT to understand the nuances of language and develop a deep understanding of context and meaning, making it well-suited for tasks such as sentiment analysis. Compared with GPT, BERT is a bidirectional model, meaning it can take into account both the preceding and succeeding context of a word when predicting the sentiment of a sentence. This with attention mechanising allows BERT to understand the full meaning of a sentence, rather than just individual words in isolation.

In our study, we established a baseline for sentiment classification using the pretrained and finetuned minBERT model on the SST and CFIMDB datasets. To extend to downstream tasks, we employed a multi-task fine-tuning model by adding a single dense layer with activation and dropout to three separate output heads.

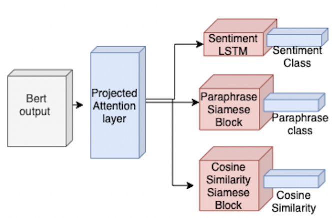
3.2 Improvements

3.2.1 Projected attention layer (PALs) and multi-head self-attention Layer

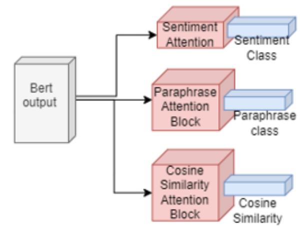
Stickland and Murray [7] proposed a method for multi-task learning using Projected Attention Layers (PALs). Our methodology investigates the variation in performance between utilizing a shared PALS layer network, which extracts common representations from multiple tasks, versus a network without shared layers consisting of only three individual task heads shown in Figure 1a. In the context of designing multi-task models, the technique of weight sharing between layers has been investigated as a crucial aspect of architecture design. During training, the model is optimized using a multi-task loss function that combines the average losses for all tasks.

3.2.2 Sentence-BERT using Siamese network

We also investigate the effectiveness of the architecture proposed by Reimers and Gurevych [8] , which utilizes a Siamese network for fine-tuning on cosine similarity. Our experiments demonstrate that utilizing the original architecture can enhance the performance of the baseline model for tasks such as paraphrasing and cosine similarity. Furthermore, we propose two improvements to the architecture to further improve its performance shown in Figure 2 . Firstly, we suggest the addition of an early u-v cross term, and secondly, we replace the pool layer with LSTM layers, which can help the model better extract features.

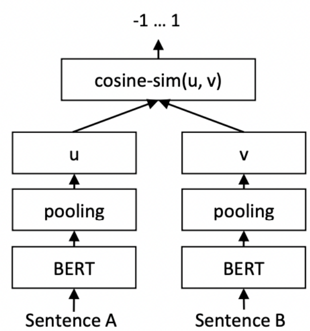


(a) Projected attention layer with shared weights

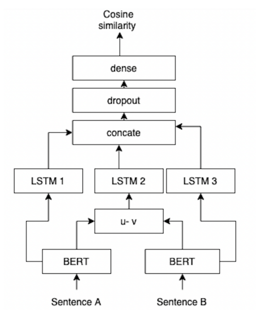


(b) Attention heads layer with no shared weights

Figure 1: Shared weights Projected attention layer



(a) Original sentence BERT architecture [8]



(b) Our designed improvement

Figure 2: Improvement over sentence-BERT architecture

3.2.3 CNN for cosine similarity

Maheshwari and Sun et al. [9] proposed an intriguing architecture for improving feature extraction in cosine similarity tasks, which involved the use of a 1D Convolutional Neural Network (CNN) as the backbone as shown in Figure 5a . This approach entailed applying ID convolutions and max pooling to individual input sentences, followed by concatenating the resulting features and passing them through a linear layer for the final output.

3.2.4 Hyper-parameter search

1. **Regression loss function** : In the context of the cosine similarity regression task, we conducted an experimental study to evaluate the performance of four different loss functions. These loss functions include the CosineEmbeddingLoss, L1 loss, Mean Squared Error (MSE) loss, and SmoothL1 loss.
2. **Multi-task loss weight** : It is well known that different tasks exhibit varying magnitudes of loss, and therefore a multi-task function that simply averages the loss values across all tasks may not be optimal. In order to address this issue, we conducted an experimental investigation into the use of different loss weight combinations for the three tasks in our study. Specifically, we utilized a methodology that assigns more weight to the smaller loss value and also more difficult individual tasks.
3. **Learning rate scheduler** : We also conducted experiments using two different learning rate schedulers, namely the linear scheduler, linear scheduler with cosine -exponential scheduler . The inclusion of cosine and exponential schedulers was motivated by the desire to achieve a steeper initial learning rate reduction during early training epochs, followed by a more gradual rate reduction during later epochs shown in Figure 5b. This approach aimed to facilitate fast convergence during initial training stages and prevent overfitting in later stages, thereby ensuring optimization of training performance.
4. **Bert layer output** :In addition, we conducted experiments utilizing three distinct output layers from the BERT model. Specifically, we investigated the performance differences

resulting from using the first token [CLS] of the last hidden state, the mean of all hidden states, and the output of the final pool layer.

3.2.5 Additional dataset

Upon conducting a thorough analysis of the training results, we observed evidence of overfitting in both the sentiment analysis and cosine similarity tasks. In response, we incorporated additional datasets for each respective task, resulting in a larger training sample size of 164,603 [10] sentiment analysis and 17,788 for cosine similarity tasks using SICK dataset.[11]

3.2.6 Ensembling

Following extensive experimentation with various architectures, we opted to utilize an ensemble method by combining the output of three best models. In the context of Paraphrase and Sentiment analysis classification task, the ensemble output is determined through a majority voting scheme. In contrast, for cosine similarity regression tasks, a linear combination of three models is utilized. As advised by Prof. Chelsea Finn in the CS330 class[12], ensemble methods are effective in mitigating the influence of random errors introduced by individual models, leaving behind only the systematic errors that can be more effectively addressed.

3.2.7 Meta-learning for sentiment classification : Proto BERT

In this study, we proposed a novel meta-learning algorithm for sentiment analysis called Proto BERT, which draws inspiration from the design of ProtoNet[13]. Our proposed sentiment meta-learning neural network architecture utilizes dense layer version as its initial design as shown in Figure 3, but can be conveniently modified to an RNN version. Additionally, we have implemented a new meta-learning data loader to enhance the efficiency of data processing.Our aim is to develop a system that can learn from a small number of examples (5 examples) , allowing it to generalize effectively to new and previously unseen sentiment analysis tasks.

Meta-learnign data loader Process The task at hand is divided into two distinct phases: Meta-learning and Meta-Test, each comprising of two separate task groups with some shared characteristics. Within D_{train} and D_{test} , the dataset is further split into support examples and query examples. At meta-learning training time, the support examples are employed to learn the model embedding, whereas the query examples are utilized to update the loss function. At the meta-test stage, a deviation from the standard supervised learning occurs in that a small proportion of 5-way or 5-support support-labelled examples are inputted into the model as support examples.

Model Process[13]
1. support examples is selected from D_{train} and processed through BERT output layers to compute the prototype of each class. The prototype is defined as the mean vector of final neural network layers for each of 5 sentiment category .

$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i)_k} f_\phi(x_i) \tag{1}$$

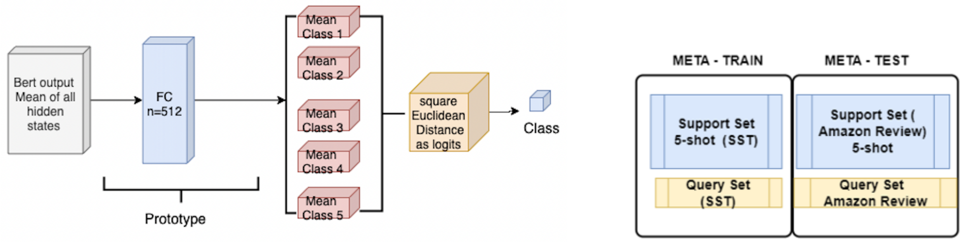
2. Then we sample some query point x, for given cluster distribution from step 1, the Euclidean distance of the query point x_i to the prototype vector is calculated. Then softmax distribution over the distance to each prototype is then determined using the following formula:

$$d(f(x), c_k) = \sqrt{\sum_{i=1}^{n} (x_i - c_k)^2} \quad and \quad p_\phi(y = k|x) = \frac{\exp\left(d(f(x), c_k)\right)}{\sum_{i=1}^{k} \exp\left(d(f(x), c_i)\right)} \tag{2}$$

3. The loss function is defined as the negative log-probability of the true class k, but only for the query examples.$-logp_\phi(y = k|x)$.

4 Experiments and Analysis

We use the datasets provided for the sentiment analysis task and multi-task fine-tuning . To perform extended experiments, our models underwent fine-tuning for 5 epochs using AdamW with a weight



(a) Proposed Meta-learning Proto Bert Architecture (b) Meta-learning data loader

Figure 3: Our proposed Meta-learning Proto Bert and dataloader

decay of 3e2. The final best results were obtained through ensembling of three models and training with learning rate decay schedules of linear-cosine-exponential for 30 epochs, using learning rates of 3e-5, 3e-4, and 4e-5.

4.1 Data

The Datasets are: Sentiment Treebank SST ,paraphrase detection ,Quora dataset , SemEval STS Benchmark Dataset. Addtional sentiment dataset[10] and STS SICK dataset [11] . And meta-learning : Sentiment Treebank SST and Amazon kindle book review Dataset. [14]

4.2 Evaluation method

Upon conducting a thorough analysis of the label distribution, we observed that there is no substantial imbalance in the classification dataset. Specifically, we did not find any class that is twice as prevalent as any other class. Consequently, we decided to solely focus on accuracy as the evaluation metric for Sentiment analysis and Paraphrase tasks. For regression task for cosine similarity , we also look at pearson correlation as evaluation metric.

4.3 Baseline

Our baseline model for single task sentiment analysis was pre-trained and fine-tuned, achieving a result of 51.8% accuracy on the SST dataset and 96.7% on the CMFIMDB dataset. For the multi-task learning approach, linear heads were added to the output of BERT, which was fine-tuned using this modified architecture. The resulting model achieved development set average accuracy results of 54%. It is possible that the multi-task fine-tuning approach did not yield an improvement in sentiment analysis development performance due to some level of overfitting.

4.4 Experimental 1: Projected attention layer (PALs) and multi-head self-attention Layer

Result As shown in Table 2, shared PALs attention was able to increase accuracy by +4.2%. By adding 1 shared attention layers helps multi-task performance , especially increase in paraphrase accuracy +12%.
Analysis Our findings suggest that incorporating projected attention layers as shared layers, along with individual task output layers, can significantly enhance the performance of the model. The additional shared weight PALs seems to help model to better balance the performance of individual tasks compared with no shared layer weights. We hypothesize that this is due to the model's ability to focus on different features that are crucial for optimal performance on each individual task.

4.5 Experimental 2:Sentence-BERT using Siamese network

Result The original design of Sentence-BERT model, which utilized a pooling layer, yielded a performance increase of +4.6% compared to the baseline. However, the results of experimenting concatenating two input sentences were not favorable and resulted in a performance decrease of -4.9%. In contrast, our proposed new architecture, which incorporates three layers of LSTM , achieved an increase of +7.3% as shown in Table 3.

Analysis After conducting extensive experiments on various architecture design choices, our findings suggest that the pooling layer may be responsible for the loss of important information. Additionally, concatenating two inputs too early without the inclusion of the u-v difference matrix may not be conducive to optimal information flow. We therefore hypothesize that replacing the pooling layer with three distinct LSTM layers (i.e., u, v, and u-v) may better extract the necessary information for the given task at hand.

4.6 Experimental 3: CNN for cosine similarity

Result As demonstrated in the Table 4, the utilization of a 1-dimensional Convolutional Neural Network (CNN) in combination with pooling resulted in an average accuracy decrease of -4.5% compared to our proposed LSTM design. It was observed that the incorporation of an additional Long Short-Term Memory (LSTM) layer prior to the CNN led to an improvement in performance, albeit still resulting in a decrease of 2.4% on average accuracy.
Analysis Our hypothesis is that the decrease in performance observed could be attributed to the localization of the feature extraction performed by the CNN. Specifically, we speculate that the kernel size of 3 may not be sufficient to capture the dependencies present in longer sentences. Thus, we conclude that a better designed CNN is required in order to surpass the performance of the Attention and LSTM-based Sentence-BERT architecture.

4.7 Experimental 4: Hyper-parameter search

Result

1. The outcomes of the regression loss function analysis indicate that the most effective approach is the Mean Squared Error (MSE) loss function, which delivers a superior performance improvement of +3.1%.Table 5
2. Furthermore, the experimentation with varying loss weights demonstrated that a weight of 1.5 times the paraphrase loss leads to an optimal performance improvement of +0.6%.Table 5
3. As shown in Figure 6, that the utilization of a linear-cosine-exponential learning rate decay scheduler results in early convergence and a performance enhancement of +0.2% over 5 epochs.Table 5 The difference could be more with more epochs of training.
4. In comparison to solely utilizing the first CLS token from the output hidden layers of BERT, taking the mean of all hidden layers leads to an improvement in performance by +1.7%.Table 5

Analysis The observed superiority of the MSE loss function in improving model performance could be attributed to the even distribution of class values within the dataset, which results in less penalization from L2 loss with outliers. Additionally, a comparative analysis of the magnitude of difference loss across three different tasks revealed that paraphrase and sentiment loss have a much smaller scale. Therefore, incorporating a weight for these losses in the model training could potentially facilitate a better balance of learning between tasks.

Based on our experimental results, it appears that the utilization of a learning rate scheduler may not be indispensable when performing fine-tuning for a limited number of epochs. However, our findings indicate that such a scheduler could potentially yield more favorable outcomes when utilized for longer training duration. Furthermore, taking the mean of all hidden layers, as opposed to solely utilizing the first token, may provide the model with more informative features.

4.8 Experimental 5: Additional dataset

Result This approach resulted in an average increase of +8.6% in the development dataset; however, the training accuracy remained at a similar level shown in Table 6.
Analysis After conducting an analysis of the baseline models for sentiment analysis and cosine similarity, we observed the presence of overfitting. Our investigation revealed a substantial dissimilarity in the quantity of training samples required for the paraphrase task in contrast to the sentiment analysis and cosine similarity tasks. Specifically, the number of samples utilized for the paraphrase task was found to be significantly greater than those employed for the latter two tasks.

Consequently, we conclude that the paraphrase task is not susceptible to over-fitting as compared to the other tasks.

4.9 Experimental 6: Ensembling Overall result

Result Shown in Table 1, ensembling led to a 2.4% enhancement in performance on the development dataset. The resulting ensemble model achieved an average test accuracy of 74%, with SST, paraphrase, and STS tasks on the test leaderboard scoring 54.2%, 81.0%, and 86.8%, respectively.

Analysis Our findings demonstrate that ensembling is a highly effective technique for improving the overall performance of machine learning models. Based on our analysis of the development set leaderboard accuracy, we have found that it is comparable to the accuracy achieved on the test leaderboard, with only a marginal difference of 1.9% attributable to model selection bias.

The ensemble model exhibits a notable improvement, with a 20.4% increase as compared to the baseline. However, there remains a significant issue of overfitting as evidenced by the sentiment analysis test results of only 54%, in contrast to the training accuracy of above 90%. For semantic similarity task, incorporating the out-of-domain SICK dataset facilitates better generalization of the model to test datasets. In the paraphrase task, the model demonstrates good training performance and better generalization to the test dataset, which can be attributed to its utilization of a large training corpus.

Ensembling	sentiment acc	paraphrase acc	semantic similarity acc	Average Acc
model1	54.9%	81.8 %	86.4%	74.1%
model2	53.2%	82.2%	86.1%	73.6%
model3	53.3 %	81.4%	86.8%	73.6%
Ensembling	55.8%	83.2%	88.9%	75.9%

Table 1: Ensembling best result for dev set

4.10 Experimental 6:Meta-learning : Proto BERT

Result As shown in the Table 7, The 5-way 5-shot few shot meta-learning result on new Amazon kindle review is 38.7 % ± 5.7% . The few shot result on the SST dev query dataset is 41.7 % ± 21.1%

Analysis We present our findings on the Amazon Kindle review dataset using a 5-shot meta-test approach. Our results indicate a promising accuracy rate of 38.7%. It is important to note that Zhong et al.[15] reported a higher accuracy rate of 46.15% for the IMDB dataset. However, it should be taken into account that the number of classes in the IMDB dataset is binary, making our task of SST 5 class sentiment analysis more challenging.Moreover, our meta-test accuracy rate is 3% lower than the 41.7% reported for the SST dataset shown in Figure 7, indicating a certain degree of overfitting to the SST domain. To improve the generalization of our model, we suggest the inclusion of more out-of-domain sentiment analysis datasets for various tasks. This can aid in expanding the model's understanding of the different nuances in sentiment analysis, leading to improved performance in diverse datasets.

5 Additional Analysis

Sentiment Class Accuracy
To investigate the underlying cause for the failure of the sentiment analysis task to demonstrate generalizability to the development dataset, despite achieving a training accuracy of over 90%. We conducted an analysis of the development class accuracy shown in Figure 4. Our analysis revealed a significant discrepancy between the class accuracy of the five sentiment classes, with classes 1 and 3 outperforming the other three classes by a large margin of 32%. Additionally, upon closer examination of the incorrectly predicted results, we discovered that a substantial majority (approximately 91.5%-95%) of these results were predicted as the adjacent neighbor sentiment class. It is possible that this

could hypothetically suggest that the model possesses some level of understanding regarding the meanings of various sentiments. However, there appears to be some ambiguity and overlap between adjacent sentiment categories, making it difficult for the model to accurately differentiate between the correct classification.

Paraphrase Accuracy
It can be seen in the Figure 4, it was observed that the task of paraphrases that were not duplicates exhibited a 9.6% higher accuracy compared to identifying duplicate pairs. Upon conducting a manual review of 20 random examples (shown 1 example below) that were wrongly classified as paraphrase, we discovered that the determination of what constitutes a paraphrase can be a subject of debate and may not always be a straightforward decision even for human evaluators.
"What are some interesting campus recruitment rejection stories?"
"What are some of the best rejection stories at campus recruitment?"

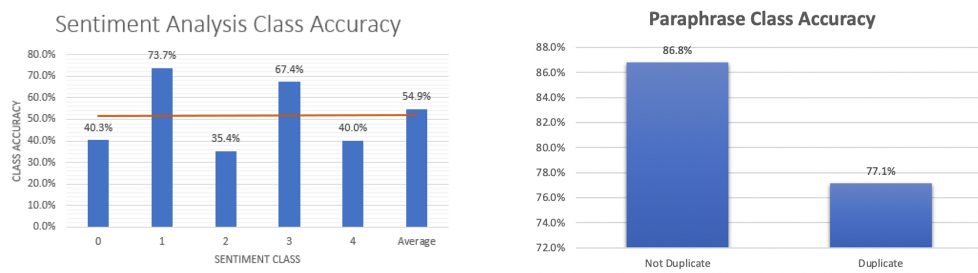


Figure 4: Sentiment Analysis and paraphrase by class accuracy

6 Conclusion

In this study, we present several key findings and improvements to enhance the performance of language models. Our results show that our approach has been successful, with an average improvement of 20.4 % in accuracy compared to baseline. Firstly, we demonstrate that the use of projected attention layers can significantly improve the model's performance. Additionally, we highlight the importance of a shared layer for multi-task fine-tuning, which can result in a significant increase in model performance. We propose an improvement to the Sentence-BERT model by introducing an early u-v difference term and replacing the pooling layer with an LSTM. These changes result in better performance on various tasks.We acknowledge that overfitting on small training datasets is a common issue in large language models. To address this problem, we recommend adding additional out-of-domain datasets to improve the model's performance on the test dataset and increase its robustness.Finally, we show that ensembling models can help reduce the random mistakes made by individual models and make predictions more consistent. Overall, our findings and proposed improvements contribute to the ongoing efforts to enhance the performance and robustness of language models.

One of the other promising extensions we have designed and implemented a Proto BERT with meta-learning, our results indicate that this approach holds great potential for the generalization of sentiment analysis tasks. Specifically, in the Amazon Kindle meta-test, the Proto BERT model achieved a promising accuracy of 38.7% when exposed to only five-shot examples.This success highlights the ability of the model to learn from limited examples and effectively generalize to new, previously unseen tasks. Our findings suggest that the Proto BERT model has promising implications for the field of meta-learning and sentiment analysis. Further research may continue to explore the efficacy and potential applications of this approach.

Our work highlights the importance of exploring and implementing various extensions in natural language processing models, as this can lead to significant improvements in their accuracy and overall performance. Additionally, our findings demonstrate the potential of meta-learning and ensemble modeling approaches for enhancing the capabilities of these models.

References

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

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[3] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

[4] Seung-Hoon Lee, Min-Ho Seo, and Jaewoo Kang. Meta-bert: Learning to learn few-shot prototypical networks for cross-domain transfer. *arXiv preprint arXiv:1912.03072*, 2019.

[5] Yixing Zhang, Wei Wu, and Tiejun Huang. Bert for cross-modal retrieval: When vision meets language. *arXiv preprint arXiv:2004.02984*, 2020.

[6] Peng Li, Peng Chen, Pei Yu, Yan Zhang, and Xiaoyan Qian. A bert-based entity linking framework with multi-grained information. *IEEE Access*, 9:27177–27187, 2021.

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

[8] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019.

[9] Danni Ma Yezheng Li Anant Maheshwari, Simeng Sun. Implementation and extensions to models in sts 2017 shared task for semantic textual similarity. `https://github.com/anantm95/Semantic-Textual-Similarity`.

[10] Baris Sayil. Sentiment analysis neural network trained by fine-tuning bert, albert, or distilbert on the stanford sentiment treebank. `https://github.com/barissayil/SentimentAnalysis`. [Version 1].

[11] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A sick cure for the evaluation of compositional distributional semantic models. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA).

[12] Prof. Chelsea Finn. (2022,24 oct), advanced meta-learning topics,cs 330: Deep multi-task and meta learning, fall 2022. `https://cs330.stanford.edu`.

[13] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. *CoRR*, abs/1703.05175, 2017.

[14] Julian McAuley. Amazon kindle book review for sentiment analysis. `https://www.kaggle.com/datasets/meetnagadia/amazon-kindle-book-review-for-sentiment-analysis`.

[15] Wanjun Zhong, Duyu Tang, Jiahai Wang, Jian Yin, and Nan Duan. UserAdapter: Few-shot user learning in sentiment analysis. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1484–1488, Online, August 2021. Association for Computational Linguistics.

A Appendix

Attention VS PALs	sentiment Acc	paraphrase Acc	semantic similarity Acc	Average Acc
dense+ dense + dense	51.1%	63.1%	81.2%	65.1%
lstm+ attention + attention	50.0%	63.0%	85.0%	66.0%
lstm+ attention + attention	52.2%	62.7%	86.0%	67.0%
attention+ attention+ attention	53.1%	62.5%	85.9%	67.2%
PAL + attention	51.6%	75.0%	81.3%	69.3%

Table 2: Projected attention layer and multi-head attention layer

Siamese network	sentiment Acc	paraphrase Acc	semantic Similarity Acc	Average Acc
Original u,v pool	51.3%	75.4%	82.6%	69.8%
1 layer lstm	51.5%	77.4%	85.6%	71.5%
1 layer lstm , cat 2 input	51.5%	66.4%	81.8%	66.6%
2 layers lstm	52.0%	76.0%	88.0%	72.0%
3 layer lstm	52.6%	79.4%	85.3%	72.4%

Table 3: Sentence-BERT using Siamese network

CNN network	sentiment Acc	paraphrase Acc	semantic Similarity Acc	Average Acc
lstm baseline	51.5%	77.4%	85.6%	71.5%
2 CNN with pool	47.0%	73.1%	80.9%	67.0%
lstm & CNN	50.0%	72.7%	84.5%	69.1%

Table 4: CNN network

Regression loss function	sentiment Acc	paraphrase Acc	semantic Similarity Acc	Average Acc
CosineEmbeddingLoss	49.8%	72.8%	47.7%	46.8%
L1 loss	47.7%	63.5%	75.0%	62.1%
L2 MSE Loss	51.1%	63.1%	81.2%	65.1%
SmoothL1Loss	48.8%	63.4%	79.3%	63.8%
Multi-loss weight				
Sentiment + 2Paraphrase + Cosine Similarity	51.3%	75.3%	84.8%	70.5%
Sentiment + 1.5Paraphrase + Cosine Similarity	52.5%	75.2%	85.5%	71.1%
1.5Sentiment + 1.5Paraphrase + Cosine Similarity	51.4%	75.1%	85.2%	70.6%
lr shedular				
linear	51.5%	62.8%	85.7%	66.7%
linear_exp_cosine	52.0%	63.0%	85.6%	66.9%
BERT output layer				
first token last hidden state	51.3%	71.8%	71.5%	64.9%
mean all_hidden_state	52.4%	75.6%	71.8%	66.6%

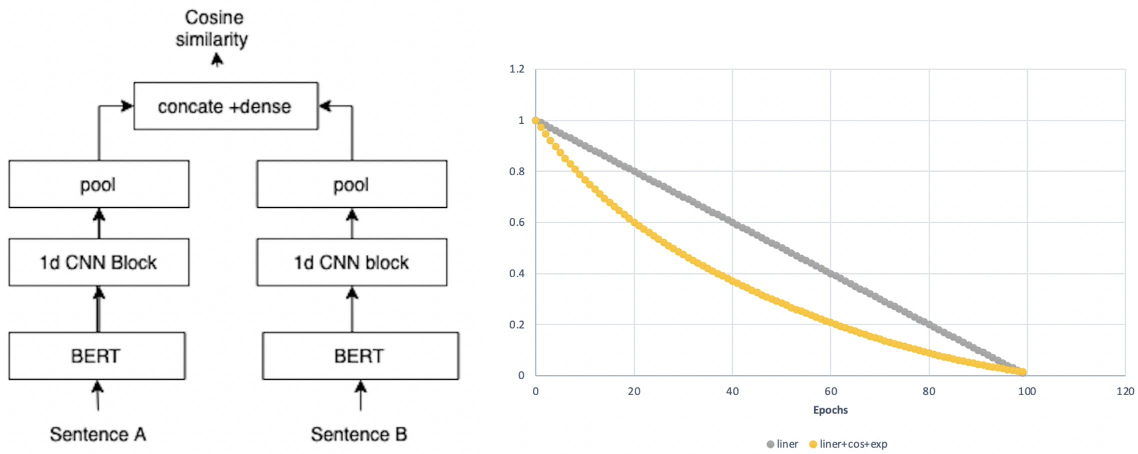
Table 5: hyper parameter search

Additional dataset	sentiment Acc	paraphrase Acc	semantic Similarity Acc	Average Acc
Default dataset	51.3%	71.8%	39.0%	54.0%
additional SST, SICK dataset	52.4%	63.1%	72.5%	62.7%

Table 6: Additional dataset

Proto - Bert	5 shot Query Accuracy
5 - way accuracy Meta- test Amazon dataset	38.7 % ± 5.7%
5 - way 5-shot accuracy SST dev dataset	41.7 % ± 21.1%

Table 7: Meta-learning query accuracy



(a) CNN architecture [9]

(b) Projected learning rate schedule

Figure 5: CNN architecture and learning rate scheduler

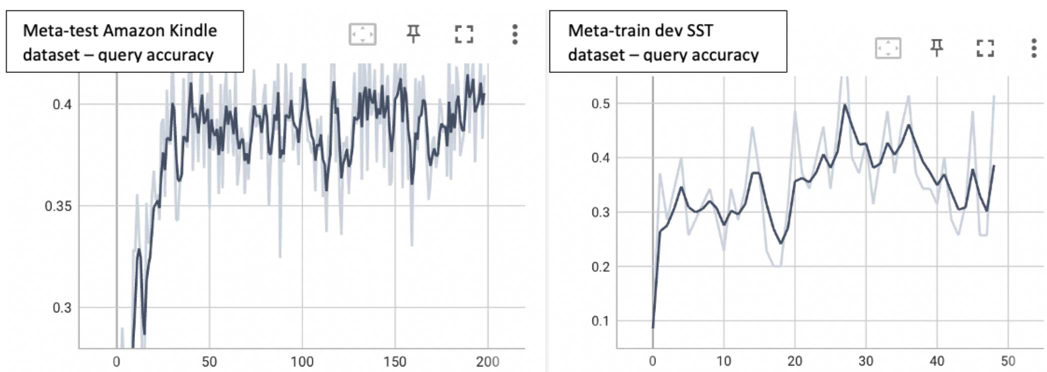


Figure 6: meta-test Amazon Review query result vs meta-train query SST result

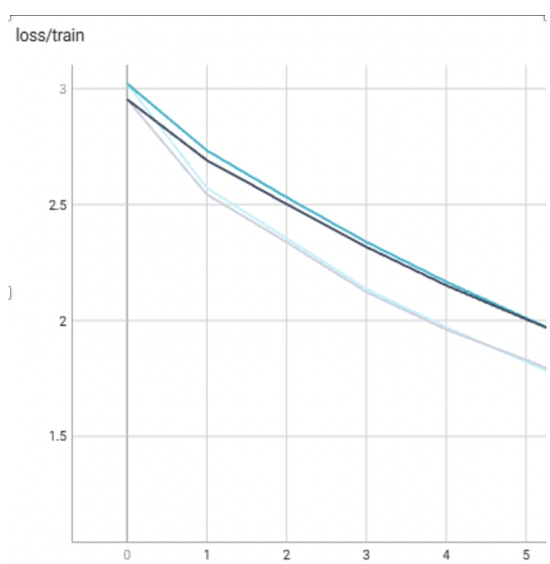


Figure 7: linear-cosine-exp scheduler early convergence with 5 training epochs