Enhancing miniBERT: Exploring Methods to Improve BERT Performance

Stanford CS224N {Default} Project Mentored by Tathagat Verma

Yasmin Salehi, Shivangi Agarwal, Ben Hora
Department of Computer Science
Stanford University
`{ysalehi, shivagar, bhora} @ stanford.edu`

Abstract

This paper investigates BERT's performance in fine-tuning tasks related to semantic classification, paraphrase detection, and semantic similarity. We propose four methods to enhance BERT: implementing a ConvBERT-inspired mixed attention block, cosine similarity fine-tuning, progressive stacking, and multitask learning. Our results indicate that incorporating ConvBERT's mixed attention block marginally improves the model's performance, which we attribute to the use of pretrained weights from BERT-base-uncased. However, employing cosine similarity during fine-tuning effectively reduces overfitting. Despite its potential, multitask learning as an additional enhancement only slightly improved the results. This lack of improvement may stem from the diverse nature of the tasks, potentially requiring more specialized model architectures or optimization strategies for simultaneous handling of multiple tasks.

1 Introduction

Recent advances in natural language processing have led to the development of powerful models such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018), which has achieved state-of-the-art results on a wide range of NLP tasks (Sun et al., 2019; Yang et al., 2019; Munikar et al., 2019; Peinelt et al., 2020; Reimers and Gurevych, 2019; Zhang et al., 2020; Jeong et al., 2020). The BERT model, originally created by Devlin et al. (2018), is designed to learn the context and meaning of words in a sentence. BERT is based on the Transformer architecture–a type of neural network that uses self-attention to model dependencies between words. More specifically, as described in Devlin et al. (2018), BERT uses a multi-layer bidirectional Transformer encoder to create contextualized word representations that consider the context of a word and its surrounding words to generate its own vector representation. In this project, we first implemented a miniBERT model including some of the key components of the original BERT model, such as multi-head self-attention and a transformer layer, and explored several methods proposed in the literature to enhance its performance. Specifically, we extended the BERT model by implementing a mixed-attention block, progressive stacking, consine similarity fine-tuning and multitask learning.

Our experimental findings reveal that incorporating the mixed attention block from ConvBERT does not lead to substantial improvements in the model's performance, which might be due to the use of pretrained weights from BERT-base-uncased. Nonetheless, applying cosine similarity during fine-tuning effectively reduces overfitting. The inclusion of multitask learning and progressive stacking as an additional modification did not result in significant enhancements in the outcomes, possibly because the tasks' diverse nature requires more specialized model architectures or optimization strategies to efficiently handle multiple tasks simultaneously. However, cosine-similarity fine-tuning effectively improved the performance by reducing overfitting and improving the contextual embeddings.

2 Related Work

Since the introduction of BERT (Bidirectional Encoder Representations from Transformers) by Devlin et al. (2018), it has become a popular pretraining technique for various NLP tasks. BERT's success has spawned several variants with modifications to improve specific aspects of the model. For instance, RoBERTa Liu et al. (2019) refines BERT by training on larger datasets and with optimized hyperparameters, resulting in improved performance. Another notable variant is ConvBERT Jiang et al. (2020), which incorporates a mixed attention block consisting of self-attention and convolutional attention mechanisms to achieve more efficient learning.

In the realm of multitask learning, BERT has been utilized for various NLP tasks simultaneously. A prominent example is the Multitask Question Answering Network (MT-QAN) by McCann et al. (2018), which fine-tunes BERT for multiple tasks, including question answering and machine translation. Another study by Stickland and Murray (2019) proposes a method for learning task-specific projection layers with BERT for multitask learning, which demonstrates improvements in model efficiency and performance.

Meanwhile, cosine similarity is a widely used metric to measure the similarity between two vectors, often employed in NLP tasks such as semantic similarity or paraphrase detection Jiang et al. (2019); Mohamed and Oussalah (2020). Fine-tuning BERT models with cosine similarity can help reduce overfitting, as it encourages the model to learn more meaningful and transferable representations. A similar approach, Siamese BERT Reimers and Gurevych (2019), uses pairwise training with contrastive loss to learn sentence embeddings, which can then be compared using cosine similarity.

Progressive stacking is a technique that involves incrementally increasing the model's capacity to enable efficient transfer learning for fine-tuning tasks Houlsby et al. (2019). This is achieved by introducing new layers or modules to the existing architecture and training them for specific tasks while keeping the pretrained weights fixed. A similar method, AdapterHub Pfeiffer et al. (2020), introduces small adapter modules in the transformer layers of pretrained models, enabling efficient fine-tuning without the need for extensive retraining. Another related approach is the use of task-specific heads on top of pretrained models, as demonstrated in Raffel et al. (2020), which allows fine-tuning for a wide range of NLP tasks.

In summary, our work builds upon these existing techniques and models to extend BERT for semantic classification, paraphrase detection, and semantic similarity tasks. By evaluating the effectiveness of mixed attention, cosine similarity fine-tuning, and progressive stacking, we contribute to the ongoing research in improving BERT-based models for various NLP tasks.

3 Approach

3.1 Mixed Attention Block

While self-attention has shown great success in NLP, it has limitations in capturing local dependencies (Jiang et al., 2020). To address this, convolutional layers may be used to achieve a better balance between capturing global context and local patterns Jiang et al. (2020). Therefore, following the work of (Jiang et al., 2020), we implemented mixed attention mechanism, which combines self-attention and convolutional layers in a single block to effectively capture both global and local dependencies (Jiang et al., 2020). The mixed-attention block is defined as:

$$\text{Mixed Attn}(K, Q, K_s, V; W_f) = \text{Cat}(\text{Self Attn}(Q, K, V), \text{SDConv}(Q, K_s, V; W_f)), \quad (1)$$

where self attention is defined as:

$$\text{Self-Att}(Q, K, V) = \text{softmax}\left(\frac{Q^T K}{\sqrt{d_k}}\right)V, \quad (2)$$

dynamic convolution is defined as:

$$\text{DConv}(X, W_f, i) = \text{LConv}(X, \text{softmax}(W_f X_i), i), \quad (3)$$

and lightweight convolution is defined as:

$$\text{LConv}(X, W, i) = \sum_{j=1}^{k} W_j . X_{(i+j-\lceil \frac{k+1}{2} \rceil}, \quad (4)$$

where $X \in \mathrm{R}^{n \times d}$ is the input and d is the hidden dimension and n is the number of tokens, and K, Q, V are the key, query and value tensors, respectively. The implementation of the mixed-attention block is illustrated in Figure 1.

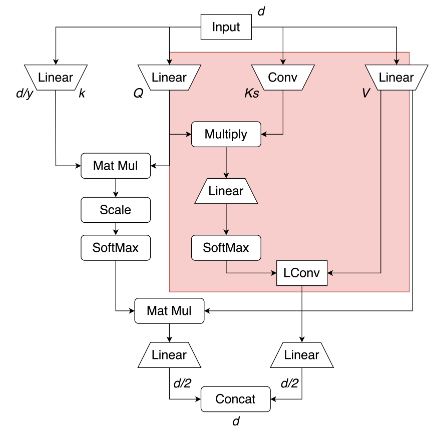

Figure 1: Illustration of mixed-attention block Jiang et al. (2020), which consists of self-attention and span-based dynamic convolution. As indicated, the modules share the same Query but use different Key to generate the attention map and convolution kernel respectively Jiang et al. (2020).

3.2 Task-specific Layers

In our project, we adopt a task-specific architecture for each of the three fine-tuning tasks, namely semantic classification, paraphrase detection, and semantic similarity. For this purpose, we implement separate layers on top of the BERT model (Devlin et al., 2018) to cater to the unique requirements of each task, which are depicted in Figure 4. In the case of semantic similarity and paraphrase detection tasks, we employ a Siamese BERT architecture Reimers and Gurevych (2019) that enables the model to learn and compare sentence embeddings effectively. This architecture consists of two identical BERT networks with shared weights, which process the input sentence pairs independently before comparing their representations.

For the semantic similarity task, we further employ cosine similarity fine-tuning Reimers and Gurevych (2019) to minimize over-fitting and enhance the model's performance. This technique involves using the cosine similarity metric during the training process to adjust the model's parameters based on the similarity between input representations. By incorporating this fine-tuning method, we aim to improve the model's ability to capture semantic relationships between sentences more effectively.

3.3 Multitask Learning

Multitask learning has been shown to be a powerful approach in natural language processing, particularly in improving the performance of large language models such as BERT. The idea behind multitask learning is to simultaneously train a single model on multiple related tasks, allowing it to learn more generalized and robust representations of the input data. By leveraging the shared representations learned across multiple tasks, the model can better capture the underlying structure of the data, resulting in improved performance on each individual task. In addition to improving performance, multitask learning can also help to reduce over-fitting and increase the model's ability to generalize to new data. Therefore, incorporating multitask learning into the training of BERT can lead to improved performance on a range of natural language processing tasks.

3.4 Progressive Stacking

The researchers in (Gong et al., 2019) introduced a novel technique for training BERT models, known as progressive stacking, which involves training the model layer by layer. This strategy allows the model to learn increasingly valuable representations of the input data throughout the training process,

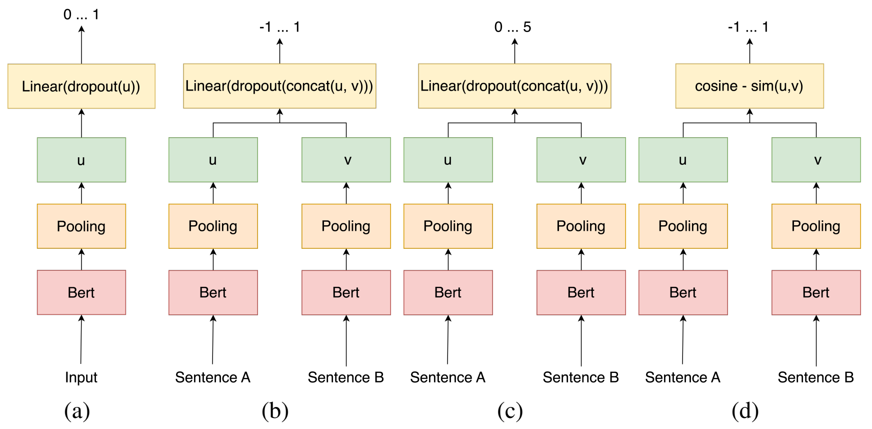

Figure 2: Task-specific layers implemented on top of the BERT model for (a) semantic classification, (b) paraphrase detection, (c) semantic similarity using neural networks, and (d) semantic similarity using cosine similarity (for fine-tuning only). As seen in the figure, (b), (c), and (d) are Siamese networks.

ultimately leading to improved performance. As a result, we integrate this method into our miniBERT model to enhance its performance. Inspired by Net2Net, a technique developed by Chen et al. (2016) that accelerates the training of neural networks through knowledge transfer, our implementation of progressive stacking follows a similar principle. Instead of transferring knowledge from a trained neural network to a larger or deeper network with similar behavior, our implementation leverages models with varying depths, such as shallow and deep models. By aggregating the knowledge from these models, we aim to create a more robust and efficient model that can potentially outperform models trained from scratch or using single-layer configurations.

We implemented progressive stacking approach by extending the original BERT model to create a ProgressiveStackingBertModel class, using an Algorithm 1 which can be found in the appendix. This class modifies the forward method to iterate through different numbers of layers specified in the self.num_layers_list attribute. For each specified number of layers, the model generates the last_hidden_state and pooler_output by calling the encode method with the specified number of layers. The outputs are then averaged across all the specified layer combinations to obtain the final output. The ProgressiveStackingBertModel aims to capture the advantages of progressive stacking as described in (Gong et al., 2019) by leveraging varying numbers of layers within the model to generate more diverse and potentially robust predictions. This approach differs from the original BERT model in that it combines outputs from different layer combinations to produce the final output, which may lead to enhanced performance on downstream tasks.

4 Experimental Settings

4.1 Dataset

The datasets on which the miniBERT are fine-tuned and evaluated on are the Stanford Sentiment Treebank (SST) and the CFIMDB datasets for the task of sentiment analysis, the Quora dataset for the task of paraphrase detection, and the SemEval STS dataset Agirre et al. (2013) for semantic textual analysis. Information regarding these datasets is presented in Table 4 in the appendix.

4.2 Pre-trained Weights

In our study, we utilize the BERT-base-uncased model as the foundation for our experiments. This model is a pre-trained version of BERT Devlin et al. (2018) that comprises 12 layers (transformer blocks), 768 hidden units, and 12 self-attention heads, totaling 110 million parameters. The BERT-base-uncased model employs the uncased variant of the tokenizer, meaning that the input text is lowercased before tokenization, and any case information is discarded. This model has demonstrated strong performance across various natural language processing tasks Devlin et al. (2018), making it a suitable choice for the foundation of our work. By using the pre-trained weights of the BERT-base-uncased model, we can leverage the knowledge it has gained from vast amounts of unsupervised pre-training on English text, allowing our model to efficiently adapt to the specific tasks of semantic classification, paraphrase detection, and semantic similarity with fine-tuning. The choice of BERT-base-uncased as the starting point for our experiments provides a strong baseline, enabling us to effectively investigate the impact of our proposed extensions and modifications on the model's performance.

4.3 Hyper-parameter Setting

Table 1 lists the hyperparameters with which different BERT extensions were trained to reach their optimal performance on the validation sets of the datasets.

Table 1: Hyperparameters

Hyperparameter	**Value**	
	pretrain	finetune
α	$1e-3$	$1e-5$
Batch size	64	32
Embedding size	768	768
Number of epochs	9	9

4.4 Model Variations

In this work, we evaluated the performance of various BERT extensions achieved by changing the convolution kernel size, k, in the mixed attention block and trying different number of layers in the progressive stacking BERT models. For proper comparison, the hyperparameters listed in Table 1 remained consistent across different experiments. The results of this ablation study are discussed in more depth in the Results and Discussion section.

4.5 Loss Function, Regularization, and Optimization

For sentiment classification and paraphrase detection, we used cross entropy and binary cross entropy loss, respectively, for learning the trainable parameters defined as:

$$H(p,q) = -\sum_i p_i \log q_i, \text{and } L(y,\hat{y}) = -\left[y\log(\hat{y}) + (1-y)\log(1-\hat{y})\right], \tag{5}$$

respectively, where p and q are two probability distributions over the same discrete set of events with p_i and q_i being the true and predicted probability of event i, y is the true binary label (0 or 1), $\hat{y}$ is the predicted probability of the positive class (i.e., the class with label 1). On the other hand, for the task of semantic similarity detection we used Mean Square Error defined as:

$$L(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

where y is the true target value, $\hat{y}$ is the predicted target value, n is the number of samples in the dataset, and L is the mean squared error loss. In our study, AdamW optimizer was used to minimize loss, and to combat overfitting, a dropout rate of 0.3 was applied throughout the architecture.

4.6 Evaluation Metrics

To evaluate the performance of our miniBERT model for sentiment classification and paraphrase detection, we used accuracy and F1-score as the primary evaluation metrics, while for semantic textual similarity, we used Pearson correlation coefficients. In this study, the results obtained by training the non-extended miniBERT model serve as our baseline.

Table 2: Pretrain Results.

Dataset	Method	Accuracy	Pearson Corr.
SST	Baseline	0.392	
	Multitask	0.388	
	Progressive Stacking	0.394	
	Mixed Attention	**0.395**	
Cfimdb	Baseline	**0.771**	
	Multitask		
	Progressive Stacking	**0.771**	
	Mixed Attention	0.735	
Quora	Baseline	**0.684**	
	Multitask	0.680	
	Progressive Stacking	0.683	
	Mixed Attention	0.677	
SemEval STS	Baseline		0.260
	Multitask		**0.261**
	Progressive Stacking		0.259
	Mixed Attention		0.288

Table 3: Fine-tune Results.

Dataset	Method	Accuracy	F1 Score	Recall	Pearson Corr.
SST	Baseline	0.523	0.522	0.531	
	Multitask Finetune	0.510	0.494	0.484	
	Progressive Stacking	**0.524**	**0.523**	**0.531**	
	Mixed Attention	0.519	0.498	0.495	
Cfimdb	Baseline	**0.967**	**0.967**	**0.967**	
	Multitask Finetune				
	Progressive Stacking	0.963	0.963	0.963	
	Mixed Attention	**0.967**	**0.967**	**0.967**	
Quora	Baseline	**0.786**	**0.769**	**0.769**	
	Multitask Finetune	**0.786**	**0.769**	**0.766**	
	Progressive Stacking	0.783	0.731	0.728	
	Mixed Attention	0.776	0.764	0.767	
SemEval STS	Baseline				0.694
	Multitask Finetune				**0.697**
	Progressive Stacking				0.662
	Mixed Attention				0.664

4.7 Results and Discussion

Main results Table 2 and Table 3 demonstrate the impact of incorporating mixed-attention mechanisms, progressive stacking, and multitask fine-tuning extensions on the performance of the extended miniBert implementations. These enhancements aim to augment the learning capability and representational power of the model. The results in Table 2 show comparable or slightly improved pre-trained weights compared to the baseline model, suggesting that these techniques enable the model to concentrate on crucial features in the input. Specifically, the mixed-attention block captures a more comprehensive range of contextual information from the input text through its combination of self-attention and convolutional layers. In contrast, progressive stacking trains the model using various layer configurations and averages the results, facilitating a gradual learning of low-level and high-level features, thus yielding marginally better outcomes.

However, while progressive stacking has led to somewhat improved performance on the SST dataset, as shown in Table 3, it has slightly overfitted to the train set of the Quora, Cfimdb, and SemEval STS datasets. This overfitting is attributable to the inherently larger number of parameters compared to the other methods and is evident through its marginally decreased performance relative to the other techniques. On the other hand, multitask learning demonstrates slightly better performance than the other approaches and the baseline on the SemEval STS and Quora datasets. We believe this outcome is due to the ability of multitask fine-tuning to reduce overfitting.

Multitask fine-tuning encourages the model to learn a shared representation that generalizes well across multiple tasks, thereby alleviating overfitting on any single task. By training on multiple tasks simultaneously, the model is less likely to overfit to the noise or idiosyncrasies in one specific dataset, as it needs to capture more general and transferable patterns to perform well on all tasks. This shared representation fosters better generalization capabilities and results in slightly improved performance on the SemEval STS and Quora datasets, as observed in the experimental results.

Effect of cosine similarity in reducing overfitting As discussed in the Methodology section, we aimed to investigate the performance of different types of layers for the task of similarity detection, namely employing neural networks and cosine similarity to predict the similarity score of two sentences. The Pearson correlation score on the validation set turned out to be 0.345 when using neural networks. In contrast, employing cosine similarity to predict the semantic similarity scores yielded better performance—i.e., a Pearson correlation score of 0.697. This significant difference in performance can be attributed to the underlying mechanisms of each method. Neural networks are powerful models that can capture complex patterns in data; however, they may require more training data and fine-tuning to achieve optimal results, or they may end up overfittig. In the case of the similarity detection task, it is possible that the neural network model may not have been adequately optimized or required more data to learn a better representation of the similarity between sentences. On the other hand, cosine similarity is a simpler, more straightforward metric that measures the cosine of the angle between two vectors. In the context of text similarity, this measure can effectively capture the semantic similarity between sentences by comparing their respective vector representations, which are typically derived from pre-trained language models like BERT. The higher Pearson correlation score achieved with cosine similarity suggests that this method is more adept at capturing the semantic similarity between sentences in the given dataset without requiring extensive fine-tuning or additional data.

Leaderboard Results Our model achieved an SST test accuracy of 0.517, a paraphrase test accuracy of 0.789, and an STS test correlation of 0.635. Combining these results, our overall test score is 0.647, placing us at position 89 in the leaderboard. These results demonstrate the effectiveness of our proposed model for semantic classification, paraphrase detection, and semantic similarity tasks. Despite the room for improvement, our position in the leaderboard indicates the potential of our approach in the context of the competitive landscape of NLP research. However, for the mixed attention block, we were expecting better results. One possible explanation for this suboptimal performance could be the use of pretrained weights from BERT-base-uncased, which may not have been optimally aligned with the mixed attention block's architecture. Additionally, the mixed attention block might not have been as beneficial for the specific tasks we addressed, given that its advantages may be more pronounced in other scenarios or domains.

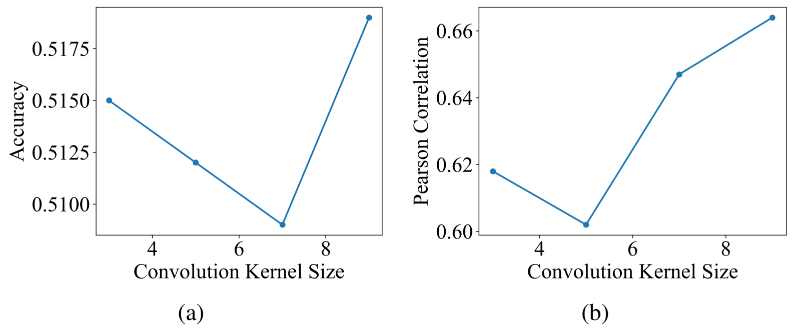

Figure 3: The effect of increasing the convolution kernel size in the mixed attention block on (a) accuracy for the sentiment classification task for the SST dataset and (b) semantic correlation for the STS dataset.

Ablation Study and Hyperparameter Search In our work, we explored the effect of increasing the convolutional kernel size in the mixed attention block, as well as various layer configurations

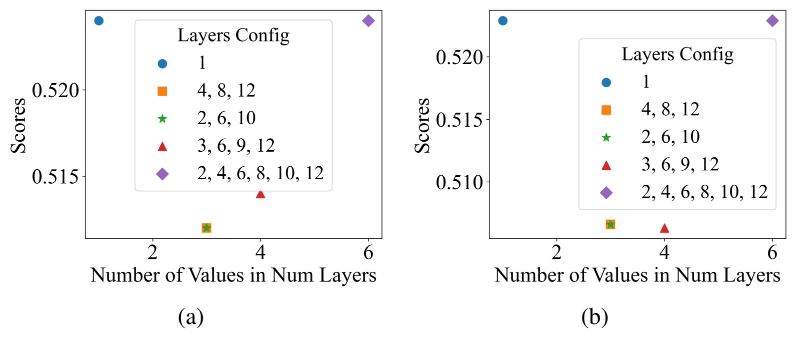

Figure 4: Comparison of progressive stacking BERT model performance with different layer configurations. (a) Development accuracy vs. number of values in Num Layers (top), and (b) F1 score vs. number of values in Num Layers (bottom).

in progressive stacking. As seen in Figure 3, increasing the kernel size of the convolution layer in the mixed attention block improved the performance on the SST and STS datasets, with $k = 9$ exhibiting the optimal performance. This observation can be attributed to the way convolutional layers operate and their capacity to capture different types of contextual information in the input text. As the kernel size increases, the convolutional layer can encompass a larger context window, allowing it to capture more long-range dependencies between words and phrases in the input text. However, we also foresee increasing the kernel size beyond a certain point to lead into diminishing returns for starting to incorporate less relevant and noisy information from distant parts of the input text. This additional information could potentially degrade the performance of the model.

Meanwhile, in progressive stacking experimentation, we analyze the results obtained from various layer configurations in the progressive stacking BERT model, as presented in Figure 3. The results indicate that the performance of the model with a single layer is on par with the performance of the model with a more layers (2, 4, 6, 8, 10, 12). This suggests that this approach of progressive stacking approach might not be as effective as initially hypothesized. Additionally, the results for the less granular configurations, such as 4, 8, 12 and 2, 6, 10, show little or no improvement over the other configurations. This could indicate that the progressive stacking approach might not provide the desired level of knowledge transfer between the layers, resulting in no significant improvement in performance. The more balanced configuration (3, 6, 9, 12) offers a slight improvement in accuracy, although the F1 score is still comparable to the other configurations. This suggests that the progressive stacking approach can yield some benefits when the increase in the number of layers is more gradual. However, the improvements observed are relatively minor and may not justify the increased complexity of the model. Overall, the results from the progressive stacking BERT model show that the approach might not significantly improve performance compared to a single-layer model. Future work could explore other configurations, different granularities, or adaptive approaches to determine the optimal layer count automatically.

5 Conclusion

The study explored mixed-attention mechanisms, progressive stacking, and multitask fine-tuning on miniBert's performance in tasks like semantic classification and paraphrase identification. Mixed-attention and progressive stacking showed marginal improvements, but progressive stacking caused overfitting in some datasets. Multitask fine-tuning reduced overfitting and slightly improved performance on SemEval STS and Quora datasets. Cosine similarity was more effective than neural networks for similarity detection tasks. Ablation analysis and hyperparameter search investigated the mixed attention block and progressive stacking, revealing that increasing the kernel size improved performance but progressive stacking did not. Key limitations include suboptimal mixed attention block performance, overfitting from progressive stacking, and modest multitask fine-tuning improvements. Future research could explore alternative configurations or adaptive methods to optimize layer count or refine the mixed attention block.

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.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. 2016. Net2net: Accelerating learning via knowledge transfer. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1–10.

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

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Efficient training of bert by progressively stacking. In *International conference on machine learning*, pages 2337–2346. PMLR.

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.

Chanwoo Jeong, Sion Jang, Eunjeong Park, and Sungchul Choi. 2020. A context-aware citation recommendation model with bert and graph convolutional networks. *Scientometrics*, 124:1907–1922.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*.

Zi-Hang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. Convbert: Improving bert with span-based dynamic convolution. *Advances in Neural Information Processing Systems*, 33:12837–12848.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Muhidin Mohamed and Mourad Oussalah. 2020. A hybrid approach for paraphrase identification based on knowledge-enriched semantic heuristics. *Language Resources and Evaluation*, 54:457–485.

Manish Munikar, Sushil Shakya, and Aakash Shrestha. 2019. Fine-grained sentiment classification using bert. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–5. IEEE.

Nicole Peinelt, Dong Nguyen, and Maria Liakata. 2020. tbert: Topic models and bert joining forces for semantic similarity detection. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7047–7055.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. 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 380–385, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. Paws-x: A cross-lingual adversarial dataset for paraphrase identification. *arXiv preprint arXiv:1908.11828*.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020. Semantics-aware bert for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9628–9635.

A Dataset Information

Table 4 includes the train:dev:test split of the datasets on which our miniBERT was trained on.

Table 4: Information on different datasets.

Name	Train	Dev	Test
SST	8,544	1,101	2,210
CFIMDB	1,701	245	488
Quora	141,506	20,215	40,431
SemEval STS	6,041	864	1,726

B Infrastructure Settings

The experiments in our study were carried on an AWS server with one NVIDIA A10G GPU, 8 CPU cores, and 23GB of memory.

C Progressive Stacking Algorithm

The following algorithm implements the progressive stacking extension in our work.

Algorithm 1 Progressive Stacking

```
Require: layer_configurations: list of lists of layer counts
Ensure: dev_accuracies: list of development accuracies for each configuration, f1_development:
    list of F1 scores for each configuration
1: Initialize dev_accuracies and f1_development as empty lists
2: for each layer_configuration in layer_configurations do
3:     Load the pre-trained BERT model
4:     Initialize the model with the given number of layers
5:     Train the model using the training data
6:     Evaluate the model using the development data
7:     Compute and store the development accuracy and F1 score in dev_accuracies and
       f1_development
8: end for
9: return dev_accuracies, f1_development
```

D ConvBERT

The SeparableConv1D class in conv_bert.py has been adapted and modified from the Hugging Face Library.