minBERT Multi-Task Fine-Tuning

Stanford CS224N Default Project

Antonio Davi Macedo Coelho de Castro
Department of Computer Science
Stanford University
`adavimcc@stanford.edu`

Abstract

This project developed a multi-task fine-tuning method for a pre-trained minBERT model to perform sentiment analysis, paraphrase detection, and semantic textual similarity tasks simultaneously. Our intuitive hypothesis believed that combining shared and task-specific parameters should result in a smaller model and superior performance compared to a set of 3 individually fine-tuned models. We propose a model architecture that combines a shared minBERT model with extra dedicated transformer layers for each task and a unified multi-task training approach that optimizes a single loss. Our results show that it is possible to achieve the same levels of accuracy as individual fine-tuned models but storing less than one-third and training less than 15% of the parameters.

1 Key Information

- Mentor: Josh Singh
- External collaborators: N/A
- Sharing project: N/A

2 Introduction

Transformer-based pre-trained language models (PLMs) such as BERT (Devlin et al., 2019), and particularly the large language models (LLMs), have demonstrated remarkable success in performing many natural language processing (NLP) tasks. However, given these models' extremely high number of parameters, fully fine-tuning them to specific downstream tasks is challenging and computationally expensive. These challenges led to a surge in the development of Parameter Efficient Fine-Tuning (PEFT) methods, which aim to reduce the number of fine-tuning parameters and memory usage while achieving comparable or superior performance to full fine-tuning. Xu et al. (2023) present a comprehensive and structured study of PEFT algorithms for PLMs. They list and classify different very successful techniques and bring the comparative results of performance experiments on benchmark datasets. We also found a promising alternative from Fu et al. (2022), in which they determined the tunable parameters using a second-order approximation method (SAM).

Despite the fast progress in developing PEFT methods, most are design-specific to fine-tune one downstream task at a time. It made us wonder: Can we fine-tune a single unified multi-task model to deliver superior performance on NLP tasks with fewer parameters? Aiming to answer this question, during this project, we developed a multi-task fine-tuning method for a pre-trained minBERT model to simultaneously perform sentiment analysis, paraphrase detection, and semantic textual similarity tasks. Our intuitive hypothesis believed that combining shared and task-specific parameters should result in a smaller model and superior performance compared to a set of 3 individually fine-tuned models. Doing this over a pre-trained minBERT model led to exciting results that we can generalize to other PLMs and LLMs. The results are auspicious, as we know that we need more effective methods to fine-tune large models to downstream tasks, given the high number of parameters.

3 Related Work

Xu et al. (2023) classify the PEFT methods for transformer-based PLMs into five different families:

- Additive Fine-Tuning: These methods introduce new trainable parameters for task-specific fine-tuning. This family has three groups. Adapter-Based, such as Pfeiffer et al. (2021), incorporates an adapter module into the transformer without modifying the pre-trained parameters. Soft Prompt-Based, such as Wang et al. (2023), appends soft prompts or prefix vectors to the input embeddings or hidden states. Others include various methods to introduce task-specific supplementary parameters during fine-tuning.

- Partial Fine-Tuning: These methods reduce the number of fine-tuned parameters by selecting critical ones and discarding unimportant ones. This family has three groups. Bias Update, such as Zaken et al. (2022), only updates the bias terms in the transformer's attention, feed-forward, and normalization layers. Pretrained Weight Masking masks pre-trained weights using different pruning criteria. Delta Weight Masking, such as Fu et al. (2022), masks via optimization approximation.

- Reparameterized Fine-Tuning: These methods utilize low-rank transformation to reduce the number of trainable parameters while allowing operation with high-dimensional matrices. This family has two groups. Low-Rank Decomposition comprehends various techniques to reparameterize the updated matrix. LoRA Derivatives are a series of methods based on Hu et al. (2021).

- Hybrid Fine-Tuning: These methods combine various approaches, such as adapter, prefix-tuning, and LoRA, to leverage the strengths of each method and mitigate their weaknesses. This family has the Manual Combination group, which combines the methods using sophisticated design, and the Automatic Combination group, which automatically incorporates the methods via structure search.

- Unified Fine-Tuning: These methods, such as Zeng et al. (2023), streamline the incorporation of many of the other fine-tuning techniques, but unlike Hybrid Fine-Tuning, they typically utilize a single method rather than a combination.

In terms of efficiency, Xu et al. (2023) present the results of extensive experiments to evaluate the effectiveness of several of these methods. Using encoder-only models RoBERTa-base (125M) and RoBERTa-large (355M) (Liu et al., 2019), which are similar to minBERT, to evaluate the GLUE benchmark (Wang et al., 2018), all PEFT methods reduce the number of trainable parameters, and most achieve performance matching or even better than full fine-tuning. The best performance was achieved with ProPELT (Zeng et al., 2023), a unified fine-tuning method, employing AdapterFusion (Pfeiffer et al., 2021) as the backbone, despite using more memory than full fine-tuning because of the extra parameters it carries on. Using encoder-decoder models T5-base (220M) and T5-large (770M) (Raffel et al., 2020) to evaluate the WMT16 En-Ro dataset (Bojar et al., 2016), LoRA (Hu et al., 2021) significantly reduces the number of trainable parameters compared to full fine-tuning while maintaining comparable performance on T5-large and superior performance on T5-base. Using decoder-only models LLaMA-7B and LLaMA-12B (Touvron et al., 2023) fine-tuned with the Alpaca dataset (Taori et al., 2023) to evaluate the MMLU benchmark (Hendrycks et al., 2021), all methods reduce the number of parameters but have an inferior performance compared to full fine-tuning.

In addition to the methods benchmarked by Xu et al. (2023), SAM is a very effective algorithm. Fu et al. (2022) propose a Second-Order Approximation Method (SAM) to better choose the tunable parameters by approximating the original problem with an analytically solvable optimization function. It is a straightforward process for selecting a small subset of the LLM parameters for fine-tuning. First, we iterate over the training dataset and calculate the initial loss gradient $\nabla_{\theta_i}\ell(\theta^0)$ for each parameter θ_i. Then, we calculate $|\nabla_{\theta_i}\ell(\theta^0)|^2$ and select the top k parameters. Despite its easy implementation, extensive experimental results show that this model outperforms many strong baseline models. In fact, during this project, we could also observe how effective this method is, and we will demonstrate that ahead.

4 Approach

Our model architecture comprises a single pre-trained minBERT model shared by the three different tasks, followed by an extra sequence of transformer layer, feed-forward, dropout, and projection dedicated to each task. For sentiment analysis, we tokenize the sentence and input it into minBERT, which outputs the pooler and the last hidden state. The last hidden state goes through one additional transformer layer dedicated to the sentiment analysis task. We then average all position encodings and concatenate the result of the extra transformer layer with the minBERT pooler output, passing it through another task-exclusive feed-forward layer. This layer comprises a dropout, linear, and GELU activation function sequence. The result then passes through a final dropout and projection layer dedicated to the sentiment analysis task, which outputs the logits for the five classes. Finally, we use the softmax function to estimate the class probabilities and the cross-entropy loss function for optimization. For the paraphrase detection task, we concatenate, tokenize, and input the sentence pair into minBERT, followed by a similar sequence of extra task-dedicated layers. The only differences are the projection layer, which outputs a single unnormalized scalar number, the use of the sigmoid function to calculate the paraphrase probability and the binary cross entropy loss for optimization. For the semantic textual similarity task, we input the sentence pair in the same way as the paraphrase detection task, and it follows the same process, outputting a single scalar number in the dedicated projection layer. We then clamp the output between 0 and 5 to calculate the similarity score and use the mean squared error loss for optimization.

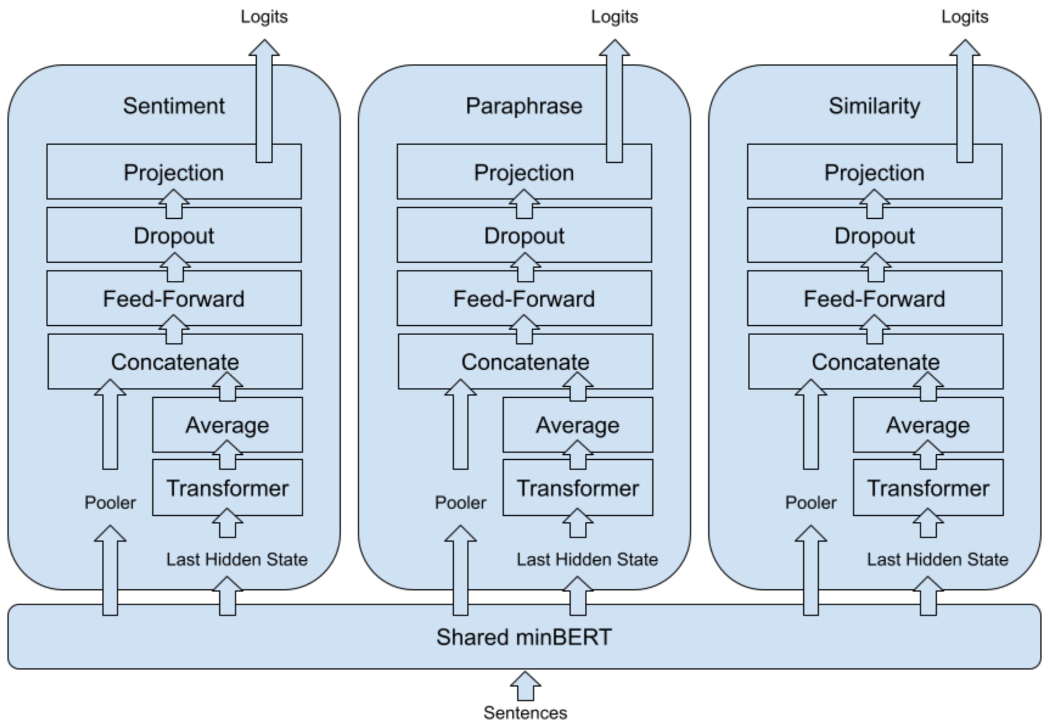

Figure 1: Model Architecture

In our methodology, we fine-tune the entire model to optimize a unified multi-task loss function $\ell_{multitask} = \frac{\ell_{sentiment}+\ell_{paraphrase}+\ell_{similarity}}{3}$. We mix the three tasks' datasets and shuffle them. In each batch, we train over a training sample that combines data from the three tasks. Since the datasets have different total sizes and we randomly select a batch, each batch can have more data from one task than another. However, we average the loss per task and finally average the three tasks' losses to calculate the final unified multi-task loss. We also make sure that, at each epoch, we go through each dataset entirely. In the first epoch, we do not optimize the model but only calculate the gradients to use the SAM method and select a k number of tunable parameters. We use the Adam optimizer based on decoupled weight decay regularization for the following iterations. It is essential to mention that

Adam does not directly apply gradients. Instead, we calculate each task's gradients separately and use the PCGrad Update Rule (Yu et al., 2020) to resolve conflicting gradients before using Adam.

5 Experiments

5.1 Data

We performed all experiments with the same training datasets. For the paraphrase detection task, we used the Quora dataset. For the semantic textual similarity task, we used the SemEval STS Benchmark dataset (Agirre et al., 2013). We used the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013) as the primary training and evaluation data for the sentiment analysis task. However, we must note that we used only a fraction of the official datasets, which made the training process harder, especially for the sentiment analysis task. We augmented the dataset we used to train for the sentiment analysis task as a solution. First, we also included the CFIMDB dataset for training. Since this dataset only contains two labels (positive and negative), different from SST, which contains five labels, during training, we transformed the five classes' logits into two class logits by doing $logits_{cfimdb}^{(0)} = logits_{sst}^{(0)} + logits_{sst}^{(1)} + logits_{sst}^{(2)}$ and $logits_{cfimdb}^{(1)} = logits_{sst}^{(4)} + logits_{sst}^{(3)} + logits_{sst}^{(2)}$. We then used the cross-entropy loss function to calculate the loss over this dataset. We also included Quora and STS for training the sentiment analysis task. For these two datasets, we assumed that if two sentences are duplicated or very similar, they must have the same sentiments. So, we isolated all sentence pairs from Quora with labels 1 (duplicated) and STS with scores over 4.5 (very similar). During training, we evaluated the sentiment probabilities for each sentence of these pair of sentences and calculated a loss using the Kullback-Leibler divergence loss function. We finally weight-averaged the loss of the four different datasets to calculate the sentiment average loss during each training batch using $\ell_{sentiment} = \frac{1}{2}\ell_{sst} + \frac{1}{6}\ell_{cfimdb} + \frac{1}{6}\ell_{quora} + \frac{1}{6}\ell_{sts}$.

5.2 Evaluation method

We first performed a full vanilla fine-tuning, individually training the model for each task to evaluate our proposed methodology. Then, we performed a second fine-tuning for each task individually using the SAM method. We used these results as our benchmarks. We finally ran our multi-task fine-tuning methodology on our unified model and compared its accuracies and number of parameters to the benchmarks.

5.3 Experimental details

Regarding hyperparameters, we used minBERT with default parameters setup, thus, a 768 hidden size and 10% dropout probability. Talking about the dedicated layers, we used the same minBERT setup for the extra transformer layer and a 1,536 hidden size for the following layers (because we concatenate the pooler with the last hidden state output average), with a 30% dropout probability. We used a 1e-5 learning rate for full fine-tuning, a 1e-4 learning rate for SAM individual fine-tuning, and our multi-task training. We fixed the weight decay at 1e-2 and used the default Adam parameters during all training. In each training, we ran the process for up to 30 epochs and selected the epoch with higher accuracy over the development dataset. We used 16 batch sizes for individual fine-tunings and 512 batch sizes for our multi-task training. We set SAM to select around 10% of the total minBERT parameters for individual training and 30% for multi-task training. After multi-task training, we performed one additional training session for each task, fixing minBERT and other task parameters and optimizing only the parameters of one task at a time. We used the previous parameters for this last training but with a 1e-5 learning rate and 16 batch size. During this final training, we did not use data augmentation and used a surprising 5e-8 learning rate for the sentiment analysis task.

5.4 Results

For our benchmarks on the development datasets, we achieved 88% accuracy on vanilla full fine-tuning for the similarity task, 89.3% for paraphrase detection, and 53.9% for sentiment analysis. These models carry together a total of 4.28e+8 tunable parameters. Using the SAM method to fine-tune each task individually, we achieved 89% for similarity, 88.3% for paraphrase, and 52.3% for sentiment, only tuning about 10% of the minBERT parameters for each task, which shows the excellent efficiency of this algorithm. The three tasks then carry together the same 4.28e+8 parameters but only tune 1.32e+8, representing 30.89% of the total.

After applying our multi-task training approach, we evaluated our trained model over the same development datasets. We achieved an 89.6% accuracy for the similarity task, 88.7% for paraphrase detection, and 51.8% for sentiment analysis. Our unified model has 1.38e+8 parameters (32.24% of the individual models), among which only 6.12e+7 are tunable (14.3% of the vanilla fine-tune models and 46.36% of the SAM individual models). After the last round of individual training, we maintained the same number of total and trainable parameters. We could improve the accuracies even further, achieving 90.2% for the similarity task, 88.9% for paraphrase detection, and 52.1% for sentiment analysis.

	Individual Vanilla	**Individual SAM**	**Unified Multi-Task**
Sentiment Analysis Accuracy	53.9%	52.3%	52.1%
Paraphrase Detection Accuracy	89.3%	88.3%	88.9%
Textual Similarity Correlation	88%	89%	90.2%
Number of Stored Parameters	4.28e+8	4.28e+8	1.38e+8
Number of Tunable Parameters	4.28e+8	1.32e+8	6.12e+7

Table 1: Benchmarks and model performance

We finally evaluated the test dataset using our best model and achieved 89.5% accuracy for the similarity task, 88.5% for the paraphrase task, and 52.4% for the sentiment task.

6 Analysis

The results indicate that our unified model and multi-task fine-tuning achieve accuracy that is very close to individual full vanilla and SAM fine-tuning on paraphrase detection and sentiment analysis tasks and is superior for the similarity task. More importantly, it achieves high gains in parameter efficiency since these accuracies only required storing less than one-third of the benchmark parameters and training less than 15

The SAM method is the main factor in achieving benchmark accuracy with fewer parameters. However, making it work was complicated since this algorithm is not designed for multi-task setup. The extra dedicated layers and the PCGrad Update Rule were crucial elements that enabled each task to perform well without interfering too much with the parameters shared with the other tasks.

Only analyzing the results achieved for the sentiment analysis task makes us believe that the effort spent on training data augmentation was in vain. Although they did not significantly increase the model accuracy on that task, after several tests, we noticed that it helped stabilize not only this task's accuracy but also the accuracy of the other tasks during training. Intuitively, since we shared training data among the three different tasks, the multi-task model was able to resolve gradient conflicts better.

7 Conclusion

We can summarize the main contributions of this project as follows:

- We designed an architecture with a shared minBERT model and additional task-specific transformer layers to perform multiple tasks simultaneously.
- We proposed a training process combining SAM and PCGrad to perform multi-task training efficiently.
- We implemented dataset augmentation methods to adapt CFIMDB, Quora, and STS datasets to train a five-class sentiment analysis task.

Achieving accuracies comparable to the benchmarks while only storing less than one-third and training less than 15% of the parameters proves it is possible to fine-tune a multi-task model sharing parameters to improve efficiency and performance. The results achieved here are fascinating and extendable to other contexts.

For future work, we want to explore how the results achieved by one task can help with the others. For example, since the similarity task achieved 90.2% over the development dataset, the knowledge acquired could be helpful for the sentiment analysis task. One possible way could be comparing the input sentence to standard negative and positive sentences.

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.

Ond rej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

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

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2022. On the effectiveness of parameter-efficient fine-tuning.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

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.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning.

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. *Journal of Machine Learning Research*, 21(140):1–67.

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.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. 2023. Multitask prompt tuning enables parameter-efficient transfer learning.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models.

Guangtao Zeng, Peiyuan Zhang, and Wei Lu. 2023. One network, many masks: Towards more parameter-efficient transfer learning.