Exploring Transfer Learning and Multi-Task Learning: An Experimental Analysis of Diverse Architectures

Stanford CS224N Default Project

Sayali Sonawane
Department of Computer Science
Stanford University
sayaliso@stanford.edu

Abstract

Transfer learning and multitask learning are pivotal techniques in deep learning, where a model is pretrained on one task and then adapted for various downstream tasks. This approach enables information sharing between tasks, significantly reducing the number of parameters needed for each individual task. In multitask learning, the overall loss is the sum of the losses from all individual tasks. As each task impacts the loss landscape of the multitask model, this study aims to identify effective methods for multitask learning, determine which methods perform well under specific conditions, and understand how different downstream tasks influence each other's performance. Additionally, we explore strategies to mitigate any negative impacts. Through research methodologies such as weight sharing, gradient surgery, LoRA and ablation studies, we analyze the performance of downstream tasks without using additional datasets, ensuring a focused evaluation of our research methodologies.

1 Key Information to include

- Mentor TA: Yuan Gao
- External Collaborators: Adyasha Maharana [2]
- Sharing project: NA

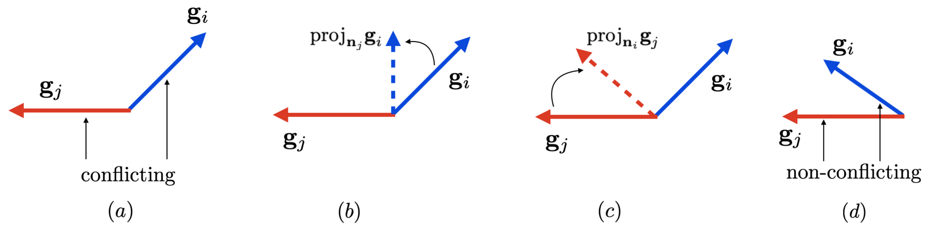
2 Introduction

In recent years, multitask learning has emerged as a powerful methodology in deep learning, where models are trained to perform multiple tasks simultaneously. This approach leverages shared representations to improve performance across all tasks by exploiting commonalities and differences between them. However, as the size of the model increases, so does the number of parameters, demanding greater computational resources to achieve optimal performance.

McCann et al developed methods often involve sharing all parameters across tasks, maintaining the same input and output spaces for each task [16]. While effective in some contexts, this approach can be limiting when tasks have different output requirements e.g. SST has 6 categorical outputs, while STS and paraphrase detection has 1.

In our research, we utilize a BERT pretrained model, maintaining shared representations through common base layers and specific linear dense layers for each task. This configuration allows us to tailor the final output layer to the unique needs of each task, facilitating better generalization and performance. By sharing the core model and dense layers, we create a robust framework

Figure 1: Projecting conflicting gradients on the normal place of the other gradient (taken from the original paper [19])



that efficiently handles diverse downstream tasks, balancing the need for shared learning and task-specific adaptations. This approach not only enhances generalization but also optimizes the use of computational resources.

To enhance the efficiency of parameter training in large language models (LLMs), we employ the Low-Rank Adaptation (LoRA) methodology [14]. LoRA utilizes rank decomposition matrices for the weight matrices, effectively reducing the number of trainable parameters. This approach allows us to train dense layers while keeping the original model weights frozen, significantly minimizing the overhead associated with task switching. Consequently, training time is reduced dramatically, as we only maintain optimizer states for the injected layers.

A notable advantage of LoRA is that it introduces no additional inference latency compared to a fully fine-tuned model. This ensures that the efficiency gains during training do not come at the cost of performance during inference. In our implementation, we inject LoRA into the query and value weight matrices with a rank of 4, optimizing the model's parameter efficiency without compromising its capability.

In multitask learning, one of the significant challenges is the optimization issue arising from conflicting gradients between tasks. According to Yu et al. [19], these conflicts occur when gradients from different tasks interfere with each other, hindering overall progress. The degree of conflict can be quantified using negative cosine similarity, which measures the extent to which gradients point in opposite directions.

To address this problem, the conflicting gradients are projected onto each other's normal plane. This projection effectively removes the interfering components, allowing the network to update its parameters without the detrimental influence of conflicting gradients. While theoretically this is possible under a certain conditions, the given algorithm doesn't work for our problem statement. The accuracy of the tasks are observed to be poor after apply gradient surgery to all 3 datasets at once. However, some increase in accuracy was observed for SST and STS datasets when GS was only applied to them.

To conduct this study of effectiveness of various architectures for multitask learning, we consider no additional datasets and compare the performance among tasks defining the baseline before implementing any additional architecture.

3 Related Work

Parameter sharing in transformer architectures has been a effective research, aimed at enhancing model efficiency and performance. The idea is explored by Vaswani et al. [18] who introduced the transformer architecture, initially focused on encoder-decoder task. Dehghani et al. [11] explored cross-layer parameter sharing in the context of fine-tuning transformers to improve language modeling. Bai et al. [9] developed the Deep Equilibrium Model (DEQ), which applies parameter sharing by maintaining the same input and output embeddings across layers. This approach achieves equilibrium with minimal oscillations, providing a stable and efficient method for deep sequence modeling. ALBERT, proposed by Lan et al. [15], achieved state-of-the-art performance by implementing cross-layer parameter sharing. We derive parameter sharing from this thoery.

Parameter optimization research on low-rank matrix decomposition provides a mathematical foundation for methods like LoRA. These techniques [17] decompose large matrices into products of smaller matrices, maintaining the essential information while reducing computational complexity. Houlsby et al. [13] introduced adapter modules for parameter-efficient transfer learning in NLP. These small, additional layers are inserted within the transformer architecture and fine-tuned for specific downstream tasks, significantly reducing the need to train the entire model. We decided to try this methodology along with parameter sharing.

Bi et al. [10] applied gradient surgery techniques to auxiliary tasks in their work on news recommendation systems. This approach involves modifying the gradients of auxiliary tasks to reduce their interference with the main task, thereby boosting the overall performance. By carefully managing the gradient contributions from auxiliary tasks, they were able to enhance the main task's outcomes while applying fewer direct modifications to main task's gradients. Their method achieves better performance through the weighted addition of gradients from two auxiliary tasks, effectively balancing their influence on the main task.

4 Approach

The goal of this study is to evaluate which transformer model architecture performs best for fine-tuning three downstream tasks: SST, STS, and paraphrase detection. We examine three different architectures: parameter sharing, parameter optimization with LoRA, and gradient surgery.

In our multitask training approach prior to parameter sharing, we combined all three datasets, calculating the overall loss as the sum of the individual losses. For inference, we added three classification heads on top of a BERT model, one for each task. Given the varying outputs of these tasks, the final dense layer was kept separate for each task, while the additional classification layers were shared among the three tasks. The model grads are reset to 0 after every grad calculation and stepped after calculating the each gradient.

LoRA layers are created as per the algorithm to calculate the value and query weight matrices with rank 4 e.g. $value\_new = value\_layer(x) + LoRA(value, x)$. These layers are then recursively inserted into the existing architecture. The weight parameters of the original model are frozen. Only the LoRA and classification layers parameters are updated.

Gradient surgery is applied during training of the multitask_classifier. The gradient of each task is calculated by individually predicting the output. Taking the pair of gradients of 3 tasks, we apply gradient surgery, where we project the conflicting gradients onto other task's normal plane. The gradients updated are then added together as per the equation $g_i = g_i - \frac{g_i \cdot g_j}{||g_j||^2} g_j$ and stepped once to update the model parameters. This reduces the influence of conflicting gradients and the influence of the larger gradients on the step function.

5 Experiments

5.1 Data

Datasets used in the project are the multitask datasets provided in the Default Project Handout [8] for the 3 downstream tasks, STS [6] for sentiment analysis, Quora datasets [7] for paraphrase detection and SST [1] for contextual similarity. All 3 datasets are used in all the experiments. These are labeled datasets, with train/dev/test division in the originally provided code.

5.2 Evaluation method

Evaluation method is common for all the experiments to efficiently evaluate the impact of individual architecture or the combination of them. An overall accuracy is calculated for all 3 tasks as provided in the DP code. The baseline is calculated from the initial experimental setup. All the additional architectures are then compared against the baseline to check their effectiveness.

5.3 Experimental details

Below is the common experiment details:

- Base Model: BERT

- Layers: 12 BERT Layers + 3 Classification Layers
- Hidden Units: 768
- Attention Heads: 8, Classification Heads: 3 (SST, STS, ParaDet)
- Classification Layers: 3 Dense Layers
- Loss Function: STS = cosine_similarity, SST and Paraphrase = cross_entropy
- Learning Rate: 1e-5
- Fine-tune Mode: full-model
- Hardware: Nvidia GPU V100 and T4, GCP
- Software: PyTorch

5.3.1 Initial Experimental Setup

For STS, initially torch.abs was used to get the difference between the labels and predicted outputs yielding poor accuracy of 0.208. Torch.abs was then replaced by cosine_similarity function improving the accuracy after model was trained on STS dataset. For Paraphrase detection, the 2 inputs were concatenated directly and trained the model with 2*hidden_size. It resulted very poor results of 0.000. It was then replaced by separated by [SEP] tokens. Accuracy was improved from 0.00 to 0.417 for baseline results. After training on all datasets it was improved to 0.553. Furthermore, the model was trained on all the downstream datasets. The accuracy improved significantly. The results are recorded in table 3. As an experiment, ReLU was replaced with GeLU but it didn't improve much results. This is considered as a baseline performance.

5.3.2 Parameter Sharing

- Parameter Sharing: Shared Classification Layers, 2 layers common among tasks, 1 output layer specific to task
- Epochs: 5
- Time per epoch: 3.5 hours

5.3.3 LoRA

- LoRA Config: Rank 4, Value and Query weight matrix, Applied to all BERT SelfAttention layers
- Epochs: 15
- Time per epoch: 2 hours
- Code referenced from TowardDataScience: Martin Dittgen [12]

5.3.4 Gradient Surgery

- GS Experiment 1: Updated gradients only for STS and SST on their respective datasets e.g. $grad_{sst} = GS(grad_{sst}, grad_{sts})$
- GS Experiment 2: Updated all gradients, e.g. $grad_{para} = GS(grad_{sst} + grad_{sts}, grad_{para})$
- Epochs: 5, 10
- Time per epoch: 4 hours
- Code referenced from github: WeiChengTseng [5] and rangwani-harsh [4]

5.4 Results

5.4.1 Baseline with Initial Setup

- Loss function for STS -> torch.abs, paraphrase input -> 2 concat matrix. Check Table 1.
- Loss function for STS -> cosine_similarity, paraphrase input -> [SEP] tokens for 2 inputs. Check Table 2.
- Training done on all datasets. Check Table 3.

Config	SST	Paraphrase Detection	STS
LL, 1e-3, dropout, 3 DL	0.398	0.00	0.083
LL, 1e-5, dropout, 2 DL	0.305	0.373	0.083
FL, 1e-5, dropout, 2 DL	0.516	0.000	0.208

Table 1: Table describing the accuracy for downstream tasks for torch.abs for STS, concat 2 inputs for paraphrase

Config	SST	Paraphrase Detection	STS
FL, 1e-5, dropout, 3 DL	0.532	0.417	0.200

Table 2: Table describing the accuracy for downstream tasks for cosine_similarity for STS, [SEP] tokens for 2 inputs for paraphrase

5.4.2 Parameter Sharing

SST dev accuracy: 0.473 (+0.170) Paraphrase dev accuracy: 0.621 (+0.068) STS dev correlation: 0.442 (-0.123).

5.4.3 LoRA

SST dev accuracy: 0.454 (+0.151) Paraphrase dev accuracy: 0.596 (+0.034) STS dev correlation: 0.725 (+0.160) Check Figures 2 and 3.

5.4.4 Gradient Surgery

Best score for Gradient Surgery achieved by updating parameters only for SST and STS.

SST dev accuracy: 0.450 (+0.147) Paraphrase dev accuracy: 0.500 (-0.053) STS dev correlation: 0.661 (+0.096) Check Figures 4 and 5.

5.4.5 Quantitative Result Analysis

- Best Test Leaderboard accuracy achieved by LoRA and shared layers: SST test accuracy: 0.488 (-0.035), Paraphrase test accuracy: 0.590 (+0.172), STS test correlation: 0.680 (+0.446), Overall test score: 0.639 (+0.120). It shows improvement of 0.120 in accuracy from the previous submission of parameter sharing. Dev accuracies are provided in the graphs attached in the appendix.
- Additional datasets for training STS and ParaDet definitely improves the performance than training on SST dataset alone, which we take as a baseline performance.
- Parameter sharing improves the performance on SST and ParaDet with 5 epochs of training, while STS accuracy decreases to 0.442 from baseline. This is expected because parameter sharing reduces the parameter count, less training required to optimize the model.
- Implementing LoRA into the existing architecture improves the performance for all the tasks. This result is submitted on the test leaderboard. LoRA performed as expected, freezing the BERT model weights and smaller rank decomposition matrices needs less training to get better results.
- Gradient Surgery overall performs very poorly than the baseline. While partial updates such as STS and SST gradients works better comparatively, SST and STS accuracies increased while ParaDet decreased. Gradient Surgery did not perform as expected. As per Bi et al. [10] updating the auxilliary gradients helped the main task, we took this approach and updated the STS and SST only to get relatively better performance.

Config	SST	Paraphrase Detection	STS
FL, 1e-5, dropout, 3 DL	0.303	0.553	0.565

Table 3: Table describing the accuracy for downstream tasks for cosine_similarity for STS, [SEP] tokens for 2 inputs for paraphrase and training done on all 3 datasets

6 Analysis

- For paraphrase detection task, the initial experiment ran with concatenated matrices. This yeilded 0 results because the concatenated matrices acted as 1 input while ParaDet needs 2 inputs to be able to compare them. Using the [SEP] token separates the 2 sentences, the model recognizes this and compares the 2 sentences for paraphrase detection, hence the accuracy increases.
- Training on one dataset vs training on multiple datasets, definitely improves performance on all tasks. More the data, better the results. Initially the model was trained on one dataset completely then on the next. This led to forgetting. After implementing the data mixing for each epoch and each step, the forgetting was less and the model improved.
- For parameter sharing, the classification layers are first kept separated to evaluate against the performance with shared layers. According to related studies, the shared learned parameters propogate the knowledge from one task to another, also known as transfer learning. The separated classification layers don't learn about other tasks, and miss out on the information which can help in other tasks.
- LoRA implements the rank decomposition matrix for training while freezing the weights of the original model. Considering the q, v matrices of size 768 x 768, it now has lesser total parameters of 768 x 4 + 4 x 768. The knowledge learned from the BERT pretraining combined with smaller fine-tuned parameters speeds up the training process and optimizes the model better. LoRA comes as blessing because lesser the learning, lesser the forgetting during multitask fine-tuning.
- Gradient Surgery performed very poorly due to the updated gradients moved away from the optimal points of the loss landscape. This is particularly not a good approach for multitask learning, because a lot of experimentation was needed to see how updating of gradients work well with the particular datasets and particular model architecture, because they define the loss landscape. The valleys of one task, might be peak of the other. Traversing the landscape efficiently is a daunting task and GS is not the best method for it. Sure, it can work really well in some cases, but overall it is observed that it is an unreliable technique to use as common methodology.

7 Conclusion

Weight sharing and LoRA achieved the best performance in this study. Gradient Surgery was up to certain extent better at achieving good results. For transfer learning and multitask classification, it can be concluded that less parameters and shared parameters for training with small data is useful approach.

The infrastructure constraints are an important issue to discuss in terms of results. Due to the limited availability of resources, such as GPUs and time, the training was limited to a few epochs. It helped saving the model state in order to continue with the training for more number of epochs.

In this study, we kept the datasets as constant i.e. studying the architecture without providing additional data. In future, I would like to explore data augmentation. With limited data, to improve the accuracy, we can use contextual embeddings, data augmentation methods [3]. Definitely adding more data is always an option to improve the accuracy.

8 Ethics Statement

We used publicly available datasets ensuring the security and privacy of the personally identifiable information. We are mindful of the ethical implications of AI technology. Our work is intended for purely academic purposes, and we caution against the misuse of our models in applications that could cause harm, spread misinformation, or violate ethical standards. We recognize the potential for bias in machine learning models, particularly in NLP tasks. We take steps to identify and mitigate any biases in our training data and model outputs by applying certain techniques like dropout. This generalizes the data enough to avoid bias.

References

[1] https://aclanthology.org/s13-1004.pdf.

[2] https://adymaharana.github.io/.

[3] https://amitness.com/posts/data-augmentation-for-nlp.

[4] . https://github.com/rangwani-harsh/pc_grad_pytorch/blob/master/pc_grad_pytorch.py.

[5] . https://github.com/weichengtseng/pytorch-pcgrad/blob/master/pcgrad.py.

[6] https://nlp.stanford.edu/sentiment/treebank.html.

[7] https://quoradata.quora.com/first-quora-dataset-release-question-pairs.

[8] https://web.stanford.edu/class/cs224n/project/default-final-project-handout-minbert-spr2024-updated.pdf.

[9] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2019. Deep equilibrium models.

[10] Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. MTRec: Multi-task learning over BERT for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669, Dublin, Ireland. Association for Computational Linguistics.

[11] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2019. Universal transformers.

[12] Martin Dittgen. https://towardsdatascience.com/implementing-lora-from-scratch-20f838b046f1.

[13] 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.

[14] 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.

[15] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

[16] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering.

[17] João G. Rosa. 2017. Superradiance in the sky. *Physical Review D*, 95(6).

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

[19] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning.

Figure 2: LoRA without parameter sharing

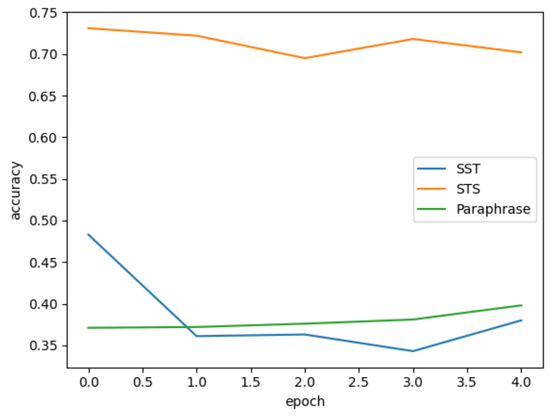
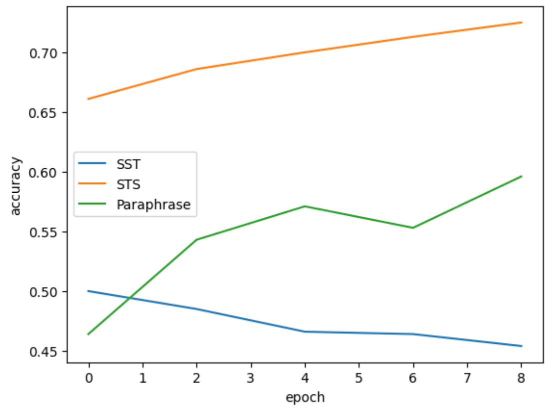


Figure 3: LoRA with parameter sharing



A Appendix

Figure 4: Gradient Surgery with parameter sharing updated params for STS and SST only

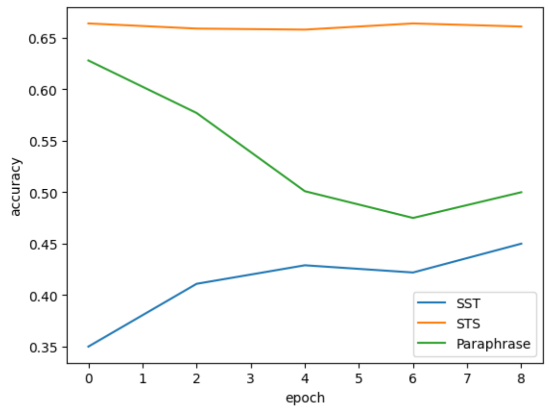
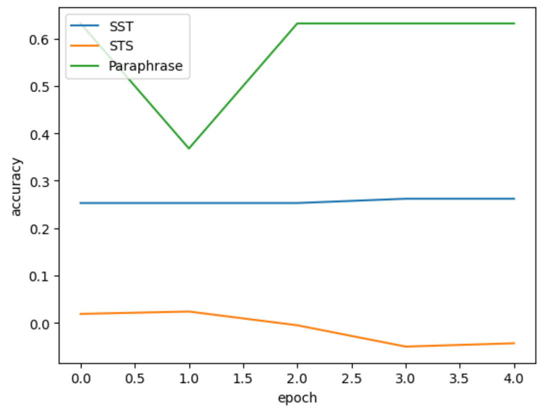


Figure 5: Gradient Surgery with parameter sharing updated params for all datasets


accuracy
epoch
SST
STS
Paraphrase