minBERT using PALs with Gradient Episodic Memory

Stanford CS224N Default Project

Christopher Nguyen
Department of Computer Science
Stanford University
cnguye29@stanford.edu

Abstract

The project aims to implement a multi-task model for the SST, QQP, and STS tasks with the main purpose of improving or matching the performance to the BERT baseline model from Devlin et al. (2019). To perform multi-task learning, the minBERT model is adapted with Projected Attention Layers (PALs) by Stickland and Murray (2019) and the training is adapted using gradient episodic memory (GEM) by Lopez-Paz and Ranzato (2017). Due to GEM's memory bottleneck, inclusion of GEM during training had worse performance compared to using an "annealed sampling" method for training with no GEM. The best model came from training without GEM which received an average development score of 0.682 and a test score of 0.690.

1 Key Information to include

- CS224n Default Project Mentor: Arvind Mahankali
- No external collaborators and not sharing project

2 Introduction

For this project, I explore how to adapt BERT to work with multiple NLP tasks simultaneously. Current approaches to learning on multiple tasks is to create separate fine-tuned models for specific tasks. Although this works better in practice, the idea of multi-tasking can help with system storage problems by reducing the number of parameters required. The motivation behind this projects multi-task learning approach is to create a more generalized model that is able to perform well across the three tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Current multi-task approaches relies on transfer learning where the model is pre-trained to learn general patterns and features useful for related tasks; but a downside to this is when fine-tuning on downstream tasks, it requires new weights for each task which can suffer from catastrophic forgetting. Forgetting weights from previous tasks can lead to worse performance. Therefore, the aim of this project is to extend BERT for the purpose of multi-tasking on a single model and find a way to mitigate catastrophic forgetting.

Extending BERT with projected attention layers (PALs) proposed by Stickland and Murray (2019) adds task-specific, low-dimensional multi-head attention layers in parallel with the existing BERT layers. To mitigate catastrophic forgetting, I attempt to incorporate Gradient Episodic Memory (GEM) by Lopez-Paz and Ranzato (2017) that alleviates forgetting while transferring knowledge to past tasks during training. To test the effectiveness of GEM I utilized two training tactics, one with the sampling method known as "annealed sampling" used by Stickland and Murray (2019) which samples tasks proportional to their training set size and evens out towards the end of training and the other with GEM utilizing a randomized round robin method.

With these extensions and modifications, the performance of BERT with PALs trained without GEM matched what was expected to BERT fine-tuned for different tasks and BERT fine-tuned jointly on tasks. The performance of training with GEM produced worse results compared to training without GEM, and did not exceed or match performance to separately fine-tuned BERT.

3 Related Work

There are many advantages to multi-task learning like transfer learning and parameter sharing. Transfer learning involves leveraging knowledge learned from one task to improve the performance on the next. Sharing parameters between tasks allows for improving model generalization by jointly learning multiple tasks. There are categories for sharing parameters, "hard parameter sharing" and "soft parameter sharing". Hard parameter sharing uses the same hidden layers for all tasks with task specific output layers. Soft parameter sharing gives each task its own model but the distances between model parameters are regularized to encourage similarity. Projected Attention Layers (PALs) (Stickland and Murray, 2019) employs the former as soft sharing requires too many parameters when using BERT. Although PALs operates by using a single model for multi-task learning and showed performance comparable to fine-tuned BERT, it still presented task interference indicated by a drop in performance on SST.

An approach used in this project to reduce interference from training on separate tasks was Gradient Episodic Memory (GEM) by Lopez-Paz and Ranzato (2017) which was brought to attention by Stickland and Murray (2019) in their paper's further discussion section but was never implemented. GEM is a model that learns the subset of correlations common to a set of tasks. GEM stores previous examples from tasks in order to alleviate catastrophic forgetting while transferring beneficial knowledge to past tasks. These stored examples are used to minimize negative backward transfer. The authors focused on computer vision not natural language processing but the premise of how GEM worked was utilized to test if a memory buffer would help with reducing task interference.

4 Approach

4.1 Baselines

Three baselines are used to evaluate the project model. The first is the GLUE benchmark test results and corresponding accuracy scores presented by Stickland and Murray (2019) which contains tasks and datasets used in this project. Given that the datasets used for testing were larger than the ones for this project, I mostly examined the accuracy trends Stickland and Murray (2019) saw when compared to the BERT-base results from Devlin et al. (2019).

The second baseline, seen as the upper bound for performance, is the implemented minBERT model fine-tuned on individual tasks and comparing its separate task accuracy results to PALs and GEM. Third baseline, the lower bound, is comparing the accuracy of a single BERT model fine-tuned jointly for all tasks. For the second and third baselines, the architecture of the BERT model was kept the same as Devlin et al. (2019).

4.2 Main Approach

The main approach is to retrofit the implemented minBERT model for PALs and adjust the training with GEM following the methods of Stickland and Murray (2019) and Lopez-Paz and Ranzato (2017) respectively. With their original open-source code as reference, I incorporated the PALs architecture and GEM algorithm into my pre-exisiting code. I referenced usage of their code in parts of my code. For the classifiers used to produce the logits for each task, I implemented an MLP with two linear layers, a ReLU activation function, and a dropout layer with probability $p = 0.1$. Additionally, I utilized the "annealed sampling" method (Stickland and Murray, 2019) and created a version of the round robin method for GEM.

4.3 Projected Attention Layers (PALs)

The objective of PALs is to incorporate a task-specific low-dimensional multi-head attention layer in parallel to each BERT layer as seen in Figure 1. The proposed layer is given by a task-specific

function (TS) of the form

$$TS(\mathbf{h}) = V^D(SA(V^E(\mathbf{h}))) \tag{1}$$

where V^E is a $d_s \times d_m$ encoder matrix and V^D is a $d_m \times d_s$ decoder matrix where $d_s < d_m$. The dimension $d_m = 768$ which is the the size of the hidden layers in the BERT model and $d_s = 204$ which is the hidden layer size of the multi-headed attention within PAL. The function takes the hidden states h to the BERT layer as input and V^E projects h to a lower dimension with a linear layer. V_E is transformed by multi-head self attention then decoded back to the original hidden size by the linear decoder layer. This allows the parameters, V_E and V_D, to be shared across layers not tasks. The parameters are added in parallel with the second residual connection ($\mathbf{h_{att}}$) and the feedforward network (FFN) before layer normalization(LN):

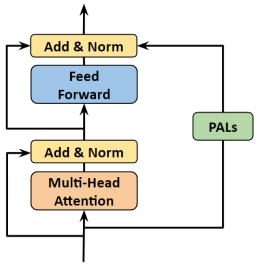


Figure 1: Schematic diagram of a single BERT layer with a parallel task-specific PAL layer

$$\mathbf{h}^{\ell+1} = LN(\mathbf{h_{att}}^{\ell} + FFN(\mathbf{h_{att}}) + TS(\mathbf{h})) \tag{2}$$

where ℓ indexes the layer. Besides the addition of a PALs layer to the BERT layers, the model follows suit with the original BERT model by stacking 12 layers on top of one another.

4.4 Gradient Episodic Memory for Continual Learning (GEM)

Gradient Episodic Memory proposed by Lopez-Paz and Ranzato (2017) is a model that alleviates forgetting from large negative backward transfer while allowing knowledge transfer to previous tasks. GEM maintains an episodic memory $\mathcal{M}_t$ that stores a subset of observed examples from each task t using integer task descriptors to index the episodic memory. The model utilizes predictors f_θ parameterized by memories from each task to ensure the loss ℓ at previous tasks don't increase after each update to allow positive backward transfer. This is done by computing the angle between the loss gradient vector of previous tasks and the proposed update. The inequality constraints is formulated as the following:

$$\langle g, g_k \rangle := \langle \frac{\partial \ell(f_\theta(x,t),y)}{\partial \theta}, \frac{\partial \ell(f_\theta, \mathcal{M}_k)}{\partial \theta} \rangle \geq 0, \forall k < t \tag{3}$$

As long as the constraints are satisfied, the proposed update is unlikely to increase the loss of previous tasks. If violations occur, a solution is to project the proposed gradient g to the closest gradient $\tilde{g}$ in squared ℓ_2 norm to satisfy the constraints. Because of this violation, we are interested in:

$$\langle \tilde{g}, g_k \rangle \geq 0, \forall k < t \tag{4}$$

To solve Equation 4, Lopez-Paz and Ranzato (2017) formulates a Quadratic Program (QP) with inequality constraints to project the proposed gradient. The primal (Equation 5) and dual (Equation 6) of the GEM QP is shown below:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} z^\top z - g^\top z + \frac{1}{2} g^\top g \\ \text{subject to} \quad & Gz \geq 0 \end{aligned} \tag{5}$$

where $G = -(g_1, \ldots, g_{t-1})$ which are the past task gradients.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} v^\top G G^\top v + g^\top G^\top v \\ \text{subject to} \quad & v \geq 0 \end{aligned} \tag{6}$$

By solving the dual problem (Equation 6) for v^*, we recover $\tilde{g} = G^\top v^* + g$ which is the project gradient update. For more detail on the GEM algorithm, see Appendix A.1.

4.5 Sampling Method

There are two types of sampling methods done in this project: "annealed sampling" and a modified round robin method. These methods were applied to the extended version of BERT that utilizes PALs and/or GEM.

Stickland and Murray (2019) constructed a sampling method known as "annealed sampling". Different from the "round robin" way to train a model on several tasks by cycling through batches of examples in a fixed order, "annealed sampling" trains tasks more equally towards the end of training where interference is more of a concern. This method reduces the effects of over-fitting on smaller tasks due to datasets in varying sizes as "round robin" will see the examples of smaller datasets multiple times before seeing every example from larger datasets. The proposed sampling method is formulated as:

$$\alpha = 1 - 0.8\frac{e-1}{E-1} \tag{7}$$

where α change with each epoch e and E represents the total number of epochs. This is put into the following equation that shows the selection of a batch of examples from task i with probability p_i at each training step and set p_i proportional to the number of training examples for a task N_i.

$$p_i \propto N_i^{\alpha} \tag{8}$$

For training with GEM, Lopez-Paz and Ranzato (2017) suggested using a more "human-like" approach where the number of training examples per task are small, the examples concerning each task are seen once and the tasks are seen in a sequential manner. Since this is essentially a round robin method, I implemented a system where each task is trained on the same number of batches and ran in a round robin fashion with random shuffling of tasks (e.g. task ids could be in order [0, 1, 2], shuffled to be [2, 0, 1], and sequentially trained on that order). This can help reduce the impact of overfitting to a certain task by ensuring that each task contributes equally to the training process. The samples in each iteration will be randomly shuffled by the DataLoader every epoch to ensure that the examples have a chance of being seen at least once.

5 Experiments

5.1 Data

I used the SST, QQP, and STS-B datasets provided by CS224N for training and testing the BERT model:

- SST: Stanford Sentiment Treebank consists of 11,855 single sentences extracted from movie reviews each has a label of negative, somewhat negative, neutral, somewhat positive, or positive. The SST is split into 8,544 train, 1,101 development, and 2,210 test examples (Socher et al., 2013).
- QQP: The Quora dataset consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another. The dataset given was a subset split into 141,506 train, 20,215 development, and 40,431 test examples.
- STS-B: The SemEval STS Benchmark dataset consists of 8,628 sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). The split is 6,041 train, 864 development, and 1,726 test examples (Agirre et al., 2013).

5.2 Evaluation method

An evaluation metrics to be used are the accuracy score for the SST and QQP tasks and Pearson correlation score for the STS task; and as stated in the baseline, I will be evaluating my extension of PALs against the accuracy and score obtained by Stickland and Murray (2019) for corresponding tasks and the accuracy of each task run on the base minBERT model with no extensions.

5.3 Experimental details

The experiment was run on the Google Console Platform deep learning VM instance with a Nvidia T4 GPU. Four methods were tested to evaluate against one other: separately fine-tuned BERT, jointly

fine-tuned BERT, extended BERT using PALs without GEM, and extended BERT using PALs with GEM. All models were trained with a learning rate of 2e-5, a batch size of 16, hidden dropout probability of 0.3, and for 10 epochs with slight variations listed below. I utilized an AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-6$, a weight decay of 0.01, and a learning rate warmup over the first 10% of steps and linear decay of the learning rate after. Each task utilized a different loss function to quantify the model's predicted outputs and the true labels. SST used a cross-entropy loss, QQP used cross-entropy with logits, and STS used mean squared error loss.

Separately Fine-Tuned BERT Used learning rate with 1e-5 for pre-training and 1e-3 for fine-tuning with no warmup and decay for each individually trained task.

Jointly Fine-Tuned BERT Trained all three tasks simultaneously with a round robin sampling method.

PALs without GEM Run with a similar PALs configuration to Stickland and Murray (2019) with twelve attention heads for the PALs layer and the annealed sampling method with 2400 steps per epoch.

PALs with GEM Used same PALs configuration with adjusted training procedure for GEM with memory size of 20, 15 epochs, and 100 batches per task.

In terms of pre-training and fine-tuning, pre-training does not require updating BERT parameters and utilizes pre-train embeddings from the BERT pre-trained model for the tasks. Fine-tuning updates the BERT parameters and obtains fine-tuned embeddings.

5.4 Results

Results of the development set for each of the four methods are presented in Table 1. The results of PALs without GEM had improvement compared to the jointly fine-tuned BERT accuracies and the QQP and STS tasks had improvement for separately fine-tuned BERT accuracies. The discrepancies between the task accuracies within the PALs without GEM method is most likely due to dataset size difference and the complexity of the SST and STS task labels in contrast to the binary labeling of the QQP task. Although the models did well in comparison to the baselines I created, both PALs without GEM and with GEM produced accuracy results lower than what Stickland and Murray (2019) received using their PALs model. The lack of training data for single sentence and sentence pair tasks may be a cause of a worse result. The difference in results may also be due to other possible factors including the usage of gradient accumulation or differences in training hyperparameters. Although the accuracies I obtained are worse than the original PALs implementation, the trends in performance for QQP was almost the exact same as BERT and a loss in performance for SST similar to what Stickland and Murray (2019) reported.

As Lopez-Paz and Ranzato (2017) applied GEM on computer vision not NLP for data and tasks, my expectations for GEM was based on the idea of how it should alleviate forgetting and direct loss to decrease without impacting the performance of previous tasks. The results on the development set showed little improvement to training without GEM. This may be due to the large memory overhead needed to store and recompute gradients which limits the episodic memory size maintained by GEM to reduce task interference.

Table 1: Development set results for the baseline models and extended models.

METHOD	SST (dev)	QQP (dev)	STS-B (dev)	Av. (dev)
Separately Fine-tuned BERT	0.515	0.780	0.384	0.662
Jointly Fine-tuned BERT	0.444	0.773	0.445	0.647
PALs w/o GEM	0.494	0.803	0.495	**0.682**
PALs w GEM	0.473	0.673	0.503	0.633

Table 2 shows comparison between PALs without GEM and with GEM on the test data. As seen from the test results, the annealed sampling method and training procedure of PALs without GEM does better most likely due to the memory overhead of having to store and recompute gradients at each learning iteration for GEM.

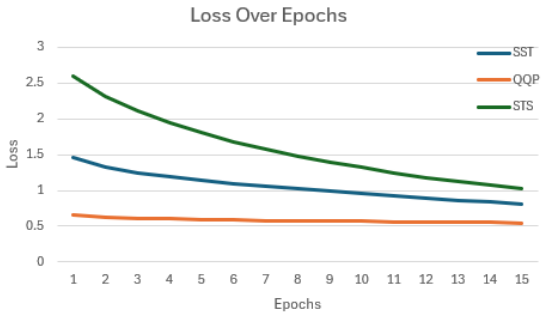
Table 2: Test set results for PALs and PALs using GEM.

METHOD	SST (test)	QQP (test)	STS-B (test)	Av. (test)
PALs w/o GEM	0.513	0.803	0.508	**0.690**
PALs w GEM	0.475	0.676	0.475	0.630

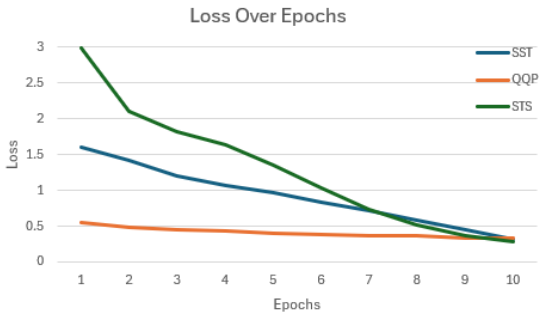
6 Analysis

From the development performance, it is evident that the task-specific layers of PALs provided better structure to BERT for multi-task learning when compared to fine-tuning for multiple tasks on a single BERT model with no modifications. The results suggest adding task-specific layers with its own attention weights helps the model focus better to relevant parts of the input sequence for each task. Additionally, separation of task-specific information in attention weights reduced interference between tasks. However when compared to the accuracies and score reported by Stickland and Murray (2019), the performance of this project's adapted BERT model did not match especially for SST where their accuracy was 93.5%.

To analyze the impact of GEM, I plotted the loss of each task every epoch for training with and without GEM shown in Figure 2. Since GEM ensures that loss of previous tasks do not increase after each iteration of a new task, the loss should be showing little variance between each epoch.



(a) GEM Training Loss



(b) Annealed Sampling Training Loss

Figure 2: Training loss every epoch for GEM and annealed sampling.

From these plots, it is clear that GEM does mitigate potential increases in loss and provides smoother convergence but the approach to how tasks are sampled leads to suboptimal performance. The annealed sampling method allows for better visibility of larger datasets which helps with not overfitting on smaller datasets. Since this method puts equal focus near the end of training, the influence of dataset size reduces task interference better resulting in a better loss reduction each epoch. It is important to note that GEM took more epochs to reach the loss the annealed sampling obtained in the first epoch for QQP which indicates underfitting.

One major point to address is how GEM is being used in this project. Lopez-Paz and Ranzato (2017) based GEM on the premise that the amount of tasks is high while the examples seen is low which differs from the project's usage of GEM. Reasons for the higher loss and lower accuracy is because GEM is bound by its memory capacity and the variance in dataset complexity and size. Firstly, GEM's bottleneck is the need to recompute previous task gradients at each learning iteration. Because of this, storing examples in memory and computing gradients for every epoch of a task slows down training time and consumes large amounts of GPU memory. Tuning hyperparameters to provide the best model while being capped by the memory capacity limits the fine-tuning of hyperparameters. GEM relies on the assumption that tasks are related and data distributions are similar enough to transfer knowledge effectively but having datasets with a ratio of the largest to smallest size of 23:1 causes performance issues for GEM. The modified round robin can lead to higher losses and subsequently lower accuracy because GEM may struggle with reducing task interference on difficult tasks whether it was a previous task or current. Annealed sampling would not improve performance as the heavy randomization of tasks and the requirement to recalculate gradients with each task change will greatly affect training time.

7 Conclusion

In this project, I implemented a modification to the original BERT model by applying PALs and attempted an approach to mitigate catastrophic forgetting during training using GEM. The model showed that PALs is an effective way to do multi-task learning and can match performance on separately fine-tuned models. However, training with GEM proved to be inefficient especially for tasks with different domains and requirements. The payoff to run GEM for reducing task interference versus the memory bottleneck was not beneficial as it limited the iterations that would have aided in learning the nuances of each task. The idea behind GEM works but may need to be modified further to work for more diverse tasks.

For future work, I want to attempt working on regularization tactics as that was not a big focus for the project. Additionally, I would like to explore different approaches to minimize task interference directly in the BERT layers using proposed method by Houlsby et al. (2019).

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.

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

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. volume abs/1902.00751.

David Lopez-Paz and Marc' Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

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

A Appendix

A.1 GEM Algorithm

Algorithm 1 Training Algorithm for GEM

$\mathcal{M}_t \leftarrow \{\}$ for all $t = 1, \ldots, T$.
$R \leftarrow 0 \in \mathbb{R}^{T \times T}$
for $t = 1, \ldots, T$ **do**
  for (x, y) in $Continuum_{train}(t)$ **do**
    $\mathcal{M}_t \leftarrow \mathcal{M}_t \cup (x, y)$
    $g \leftarrow \nabla \ell(f_\theta(x, t), y)$
    $g_k \leftarrow \nabla \ell(f_\theta, \mathcal{M}_k)$ for all $k < t$
    $\tilde{g} \leftarrow$ **PROJECT**$(g, g_1, \ldots, g_{t-1})$ ▷ See Eq.6
    $\theta \leftarrow \theta - \alpha \tilde{g}$
  end for
  $R_t :\leftarrow$ **EVALUATE**$(f_\theta, Continuum_{test})$
end for