Multi-task Fine-tuning with BERT

Stanford CS224N { Default} Project

Sanjaye Elayattu
Department of Computer Science
Stanford University
sanjaye@stanford.edu

Abstract

Single task natural language processing (NLP) models have a single transformation layer on top of a large language model(LLM) such as BERT. This combined model is then trained to perform the specific NLP task. However, the LLM models use large number of parameters. The base BERT model uses approximately 110 million parameters and the recent LLM model GPT-3 has 175 billion parameters.

Multi-task NLP models are built to benefit from transfer learning and parameter sharing. In this scheme, a multi-task model is built using a LLM as a base. It performs multiple tasks using task specific transformation that are added to the "top of the LLM model" or "within the LLM model" or a combination thereof. These models beneift from transfer of knowledge due to training with multiple, related NLP training data and also share all the parameters of the base LLM.

In this project, we consider three NLP tasks using a Multi-task model that uses BERT as a foundation. We consider three tasks; Sentiment Analysis(SST), Paraphrase Detection(Para) and Semantic Textual Similarity (STS).

We explore two design choices for the model architecture. 1) The baseline model simply adds a transformation for each tasks on top of the BERT[1] model based on the sentence representations provided by BERT. 2) We also experiment by creating three Projected Attention Layers(PALs), one for each task, which share weights across layers (not across tasks). The goal of this approach is to reduce the number of additional parameters required compared to task-specific models and also realize better transfer learning due to the model architecture as well as training methods.

We evaluate the baseline model with pretrained and fine-tuning approaches and observe that the multi-task model needs to be fine-tuned on all three datasets. Multi-task learning often comes with the caveat that size of the available training data can be different for various tasks. We attempt to avoid over-fitting for tasks with smaller data sets by using anneal sampling of training data as against round robin method

1 Key Information to include

- Mentor: N/A
- External Collaborators (if you have any):N/A
- Sharing project:N/A

2 Introduction

To create a baseline, we implement mini-BERT which provides representations for sentences as part of its final output. We then create a multitask classifier which performs the 3 task-specific transformations on the output representation provided by the BERT model.

First, we evaluate the baseline model using pre-training [1] and fine-tuning techniques and observe that fine-tuning on task-specific training data provides better accuracy on the specific task compared to just using pre-trained BERT weights as inputs to task specific transformations.

Second, we continue fine-tuning the model with different training data sets and sampling methods. We observe that the models fine-tuned specifically on certain training data tends to provide better accuracy on the corresponding task but does not generalize well to other tasks. We also note that fine-tuned model contains 110 million parameters from the BERT model plus additional parameters required for task specific transformations. We expect that Multi-task learning will not only help reduce the number of parameters but also help with model generalize across the tasks by sharing parameters.

As a next step, we fine-tune the model on all the three training data sets. This approach provides reasonable accuracy on all three tasks and does not require large number of parameters. We also observe that round-robin sampling of training tends to over fit tasks with smaller training datasets. We also notice that anneal sampling helps the model generalize well and avoid overfitting, when compared to round robin sampling of the training data set.

Finally, we add Projected Attention layers for each task, within the 12 BERT layers that share parameters across the BERT layers(not across tasks). We observe that this approach allows us to provide task specific learning within the BERT layers while only adding relatively smaller number of parameters(apprximately 1.9 million per task) to the model. We observe this approach provides accuracy results comparable to that of single task fine-tuning.

3 Related Work

Deep neural networks in general have limitation that they use a large number of parameters to perform a specialized tasks. Rebuffi et al.[2], argue that transfer learning techniques which uses fixed features provided the large model as the input to the task specific model may not perform as well as specialized neural networks. Rebuffi et al.[2] propose an approach to modify the model with some task specific parameters to achieve the balance between generalization across multiple tasks and task specific accuracy.

Copper et al.[2] propose projected attention layers; a multi-task learning technique that reduces the number of additional parameters significantly. Copper et al[2] explore several approaches such as adding parameters to the top of the BERT layer, adding new BERT layer per task, Projected Attention Layers(PALs)[2] and compare their model performance against the GLUE [3] benchmarks. They key observation from this paper is that, by using adaptive multi-task learning using PAL, the multi-task model provides comparable performance against GLUE benchmarks by using significantly lower number of parameters.

4 Approach

4.1 Baseline Multi-task Model

The following diagram shows the multi-task model that uses an implementation of BERT, we call it mini-BERT. The input to the BERT model is sentence pair, converted into word tokens. The input is a total of 512 tokens including the CLS token that marks the start of the sentence and the SEP.

The embedding layer in the mini-BERT is a summation of word embeddings (512 tokens), position embedding (512 positions and token type embeddings(2 token types). The embedding dimension is 768. Each BERT Layer consists of multi head attention[4] with a residual connection to the hidden

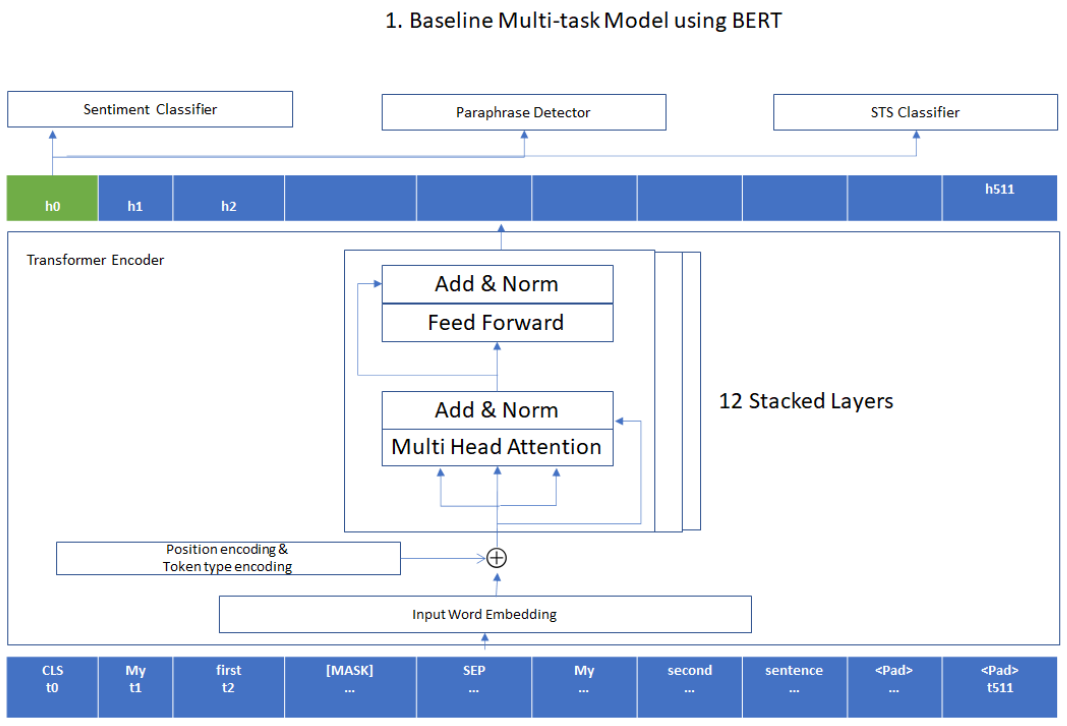


Figure 1: Multi-task model with classifiers on top

inputs. This interim output is then fed into a feed forward network with a residual connection. Our mini-BERT model also uses 12 such BERT layers stacked on top of each other to complete the transformer encoder architecture[4].

We use the output vector corresponding to the first hidden layer (h0 in the diagram) as representation of the input sequence. This input is then fed as input into the task-specific transformations for Sentiment Classification, Paraphrase Detection and STS classification.

The task specific transformations are applied at the top of the model leaving the basic mini-BERT model untouched. Following table describes the task specific transformations.Figure 2 shows the inputs, transformations and loss functions for each of the task specific transformations.

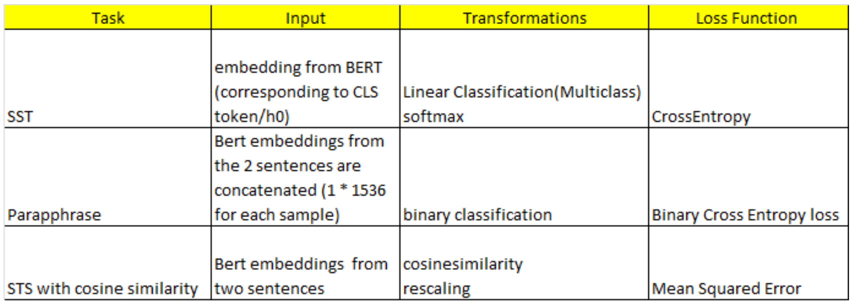
For Sentiment Classification (SST), we simply perform a multi-class classification use Crossentropy function to evaluate the losses.

For paraphrase detection there two input sentences. The BERT embedding for these two sentences are concatenated into a single vector per input sample (dimension=1536). Binary classification output is evaluated using binary cross entropy loss.

For Semantic Text Similarity (STS) classification, we tried two approaches. We initially tried tried linear regression using the concatenated BERT embedding for the 2 input sentences(dimension=1536). Later we modified this to classification to use, cosine similarity between the two input vectors(as shown below). The output from the cosine similarity is in the range of -1 to 1, which was rescaled into a range between 0 and 5 to align with the input labels.

4.2 Multi-task Model with PAL

In this model, we modify the mini-BERT layers such that one Projected Attention Layer is added for each task within the BERT layers. T.



Task	Input	Transformations	Loss Function
SST	embedding from BERT (corresponding to CLS token/h0)	Linear Classification(Multiclass) softmax	CrossEntropy
Parapphrase	Bert embeddings from the 2 sentences are concatenated (1 * 1536 for each sample)	binary classification	Binary Cross Entropy loss
STS with cosine similarity	Bert embeddings from two sentences	cosinesimilarity rescaling	Mean Squared Error

Figure 2: Brief description of the 3 task specific transformations

In the paper about Projection Attention Layers, Cooper et.al[2] discusses few options for adding parameters within BERT. The task specific functions are of the form

$$TS(h) = V^D g * (V^E h) \tag{1}$$

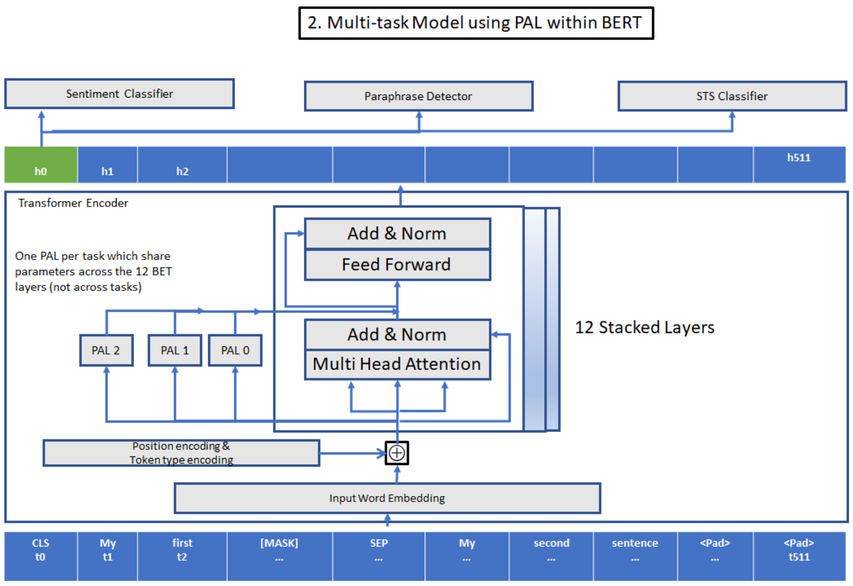


Figure 3: PAL layers added for the 3 three tasks within BERT layer

Here TS is the task specific function based on the hidden state output from BERT. V^E is the encoder matrix with a dimension of $ds * dm$

and V^D is the decoder matrix with a dimension of $dm * ds$ The function $g(.)$ can be an identiy function, multi-head attention or a feed forward neural network. For our experiments we used $g(.)$ as a multi-head attention with V^E and V^D shared across the 12 BERT layers. This is illustrated in Figure 3 using the boxes named as PAL. Note that the parameter sharing is across the 12 BERT layers; they are not shared among the three tasks under consideration. In this equation d_m is the dimension of the BERT embedding (768) and d_s is chosen to be 204 to reduce the number of additional parameters added. Since V^E and V^D are shared acorss tasks and multi-head attention will require 3 times $(d_s)^2$ parameters, the total number of additional parameters added is

$$T * (2 * d_m * d_s + 12 * 3 * (d_s)^2) \tag{2}$$

, where T= number of tasks (3 in our case), formula excludes bias parameters.

The rest of the multi-task model architecture is same as that of the baseline Multi-task model shown in Figure 1

5 Experiments

The following types of experiments were ran. a) Pretrained vs Fine Tuning b) Sequential / Round Robin Sampling

5.1 Data

5.1.1 Sentiment Analysis data set

For the sentiment classifier we use the Stanford Sentiment TreeBank(SST) which has 11,855 movie reviews rated as negative, somewhat negative, neutral, somewhat positive or positive (corresponding to the logits(0-4) used in the classifier)

Movie Review: Light, silly, photographed with colour and depth, and rather a good time.
Sentiment: 4 (Positive)
Movie Review: Opening with some contrived banter, cliches and some loose ends, the screenplay only comes into its own in
Sentiment: 2 (Neutral)
Movie Review: ... a sour little movie at its core; an exploration of the emptiness that underlay the relentless gaiety of the 1920's ... The film's ending has a " What was it all for?"
Sentiment: 0 (Negative)

5.1.2 Paraphrase Detectiondata set

For the paraphrase detection we use a subset of the Quora data set with 202,152 paraphrase examples. These are paris of sentences labelled with 0 or 1 indicating whether the second sentence is a paraphrase of first one, as shown in examples below

Question Pair: (1) "What is the step by step guide to invest in share market in india?", (2) "What is the step by step guide to invest in share market?"
Is Paraphrase: No

Question Pair: (1) "I am a Capricorn Sun Cap moon and cap rising...what does that say about me?", (2) "I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?
Is Paraphrase: Yes

5.1.3 Semantic Text Similarity(STS) data set

For Semantic Text Similarity (STS), we use the SemEval STS benchmark with 8628 examples. These are also pairs of sentence with the label indicating whether the two sentences are semantically similar. The labels range from 0 to 5 as described below with examples

5) The sentences are completely equivalent, as they mean the same thing:
The bird is bathing in the sink.
Birdie is washing itself in the water basin

(4) The two sentences are mostly equivalent but some unimportant details differ:
In May 2010, the troops attempted to invade Kabul.
The US army invaded Kabul on May 7th last year, 2010.

(3) The two sentences are roughly equivalent, but some important information differs:
John said he is considered a witness but not a suspect
"He is not a suspect anymore."

(2) The two sentences are not equivalent, but do share some details:
They flew out of the nest in groups.
They flew into the nest together.

(1) The two sentences are not equivalent, but are on the same topic:
The woman is playing the violin.
The young lady enjoys listening to the guitar

(0) - Unrelated;
Tomorrow is Monday.
Lion is a carnivore.

5.2 Evaluation method

For the Sentiment Analaysis and Paraphrase detection, we use the accuracy scores as a way to evaluate the model performance. For the Semantic Text Similarity, we use the pearson correlation as a score to evaluate the model performance.

5.3 Experimental details and Results

1.Finetuning Pre-training against different datasets

The figure 4 below shows the first set of experiments to create a baseline multi-task model. The first set of experiments are listed below. Experiment 1 used training data from all three tasks, where experiment 2, 3 and 4 used only one of the data sets. In experiments 2, 3 and 4, the accuracy results are better on the tasks whose training data was used at the expense of poor accuracy scores on other tasks.

In all four experiments, the model was evaluated on development data using pre-trained and fine-tuning approaches. Fine-tuning consistently over-performed training pre-training approach in terms of accuracy scores. In all cases, pre-training approach was faster than finetuning in terms of training time.

In all these experiments, all the training data was fed sequentially into the model on each epoch.

The model form experiment1 served as baseline for the rest of the experiments.

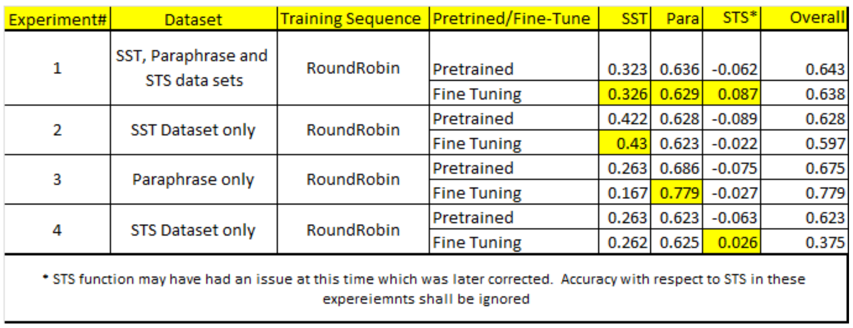
2.Anneal Sampling vs Round Robin Sampling

Experiments 1 through 5 uses Round Robin sampling where the training loop goes through each of task specific data sets in a sequence. Since we paraphrase data set contains approximately 18 to 20 times that of Sentiment(SST) and Semantic Text(STS) data sets, this training method can potentially over-fit SST and STS training data resulting in poor test performance. Figure 5 shows the details about experiments 5 through 9. In experiment 6, the sampling method was switched over to Annealed sampling [2]. In this method, the batches are picked based on a probability distribution given by

$$p_i \propto N_i^{\alpha} \tag{3}$$

where N_i is the number of examples available for each task and α changes on each epoch e (E being the total number of epochs).

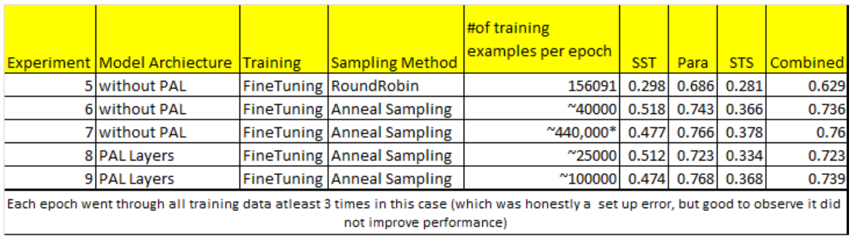
$$\alpha = 1 - [0.8 * (e-1)/(E-1)] \tag{4}$$



Experiment#	Dataset	Training Sequence	Pretrined/Fine-Tune	SST	Para	STS*	Overall
1	SST, Paraphrase and STS data sets	RoundRobin	Pretrained	0.323	0.636	-0.062	0.643
			Fine Tuning	0.326	0.629	0.087	0.638
2	SST Dataset only	RoundRobin	Pretrained	0.422	0.628	-0.089	0.628
			Fine Tuning	0.43	0.623	-0.022	0.597
3	Paraphrase only	RoundRobin	Pretrained	0.263	0.686	-0.075	0.675
			Fine Tuning	0.167	0.779	-0.027	0.779
4	STS Dataset only	RoundRobin	Pretrained	0.263	0.623	-0.063	0.623
			Fine Tuning	0.262	0.625	0.026	0.375

* STS function may have had an issue at this time which was later corrected. Accuracy with respect to STS in these expereiemnts shall be ignored

Figure 4: pre-training vs Finetuning with RoundRobin sampling in model without PAL



Experiment	Model Archiecture	Training	Sampling Method	#of training examples per epoch	SST	Para	STS	Combined
5	without PAL	FineTuning	RoundRobin	156091	0.298	0.686	0.281	0.629
6	without PAL	FineTuning	Anneal Sampling	~40000	0.518	0.743	0.366	0.736
7	without PAL	FineTuning	Anneal Sampling	~440,000*	0.477	0.766	0.378	0.76
8	PAL Layers	FineTuning	Anneal Sampling	~25000	0.512	0.723	0.334	0.723
9	PAL Layers	FineTuning	Anneal Sampling	~100000	0.474	0.768	0.368	0.739

Each epoch went through all training data atleast 3 times in this case (which was honestly a set up error, but good to observe it did not improve performance)

Figure 5: Experiments with different sampling methods, epoch sizes and the 2 models (PAL and without PAL

Intuitively, this equation allows choosing training data volumne proportional to the size of the datasets in the early epochs and choosing the training data volume almost equally for all tasks towards the last few epochs.

Switching from round-robin to annealed sampling not only provided better accuracy/scores across all the tasks; but it also allowed to have a smaller training data set to be used for each epoch. This was managed by new parameter "batches Per epoch" , managed by a new training parameter.

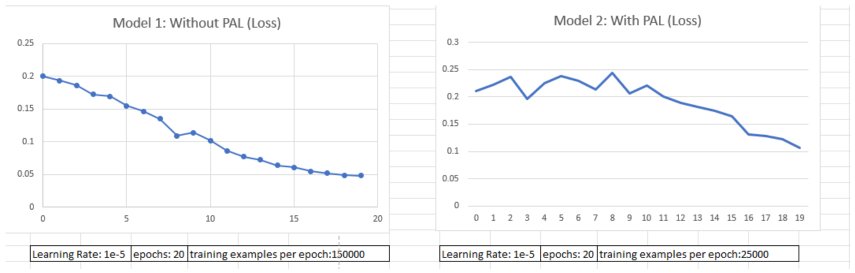
We ran one more experiment, 8, with a really large number of examples per epoch (this was done accidentally and abandoned after 8 epochs). However, the results were inconclusive as to whether more training epochs or batches per epoch would have improved the scores. The accuracy scores in 6 and 7 are comparable even though experiment number 7 would take almost 24 to 26 hours to complete.

3.Model Comparison: without PAL and with PAL Experiments 8 and 9 can be compared with that of experiment 6 to see the difference in model performance for the original model without PAL and the model with PAL. Based on the results of these experiments, these models provide comparable performances. for the same number of epochs and similar settings of hyper parameters.

The following diagram shows the loss curves for the two models. Based on the loss digaram it appears that the model with PAL may take little longer to find the point of minimal loss. While this is possibly justified due to the increased number of parameter due to addition of PAL layers within the BERT layer (with shared parameters across the BERT layers and unique parameters across the tasks), we have not done enough experiments to conclude whether the PAL model performs better than the one without PAL with more training. We leave this topic for a future research.

Hyper Parameters setting We used a batch size of 64 for Sentiment and Semantic text data and used a batch size of 32 for par phrase training in all the experiments. We used hyper parameter of 1e-5 for all fine-tuning tasks and a learning rate of 1e-3 for all pretrainng tasks in experiments 1 thorugh 9. We used Adam optimizer with a learning rate of 1e-5 $\beta_1 = 0.9$, $\beta_2 = 0.999$ and L2 weight decay of 0.01

Leaderboard Submission Results Following are my DEV and TEST leaderboard submission results.Rest of the experiment details are already provided in the above section.



Learning Rate: 1e-5 | epochs: 20 | training examples per epoch:150000

Learning Rate: 1e-5 | epochs: 20 | training examples per epoch:25000

Figure 6: Loss at each epoch for the 2 models without PAL and with PAL

Leaderboard	SST Accuracy	Paraphrase Accuracy	STS Correlation	Overall
DEV	0.474	0.753	0.363	0.530
TEST	0.512	0.754	0.326	0.531

I believe the results are on par with my expectations. The focus on my experiments was to get the basic mini-BERT model right, optimize the training method and also modify the mini-BERT to implement PAL. With some additional experiments with hyper parameter chagnes and further analysis, I believe the model performance can be improved. I also believe the training would be much better if we have a larger data set similar to that used in paraphrase task for both sentiment and semantic text tasks.

6 Analysis

The mini-BERT model that we created use close to 110 million parameters. Multi-task learning allows us to share these parameters across different tasks while using additional parameters that are used for task specific transformations. By training the BERT model on multiple tasks, the 110 million parameters are expected to generalize better. If we create one model per task, the fine tuned models will use parameters in the order of 110 Million.(110 Million plus additional parameters required for specific task).

The two models provided use significantly less number of parameters due to the sharing of BERT parameters, fine tuned by the various training tasks. Model 1 adds task specific transformation to the top of the BERT. Model two basically widens the BERT (due to the PAL layers within BERT layers) but still uses less number of parameters when compared with task-specific models. We expect the PAL model which adds approximately 1.9 million additional parameters per task to the 110 million BERT parameters to perform better due to its interaction with other parameters within the BERT at each layer.

7 Conclusion

These experiments also helped to understand the efficiency of just using the pretrained model as well as the benefits of finetuing the model for better scores. We were also able to clearly see the advantage of using annealed sampling as against round robin sampling to not only see better accuracy scores but to also attain better performance. In the paper on PALs[2] Cooper et.al discussed several approaches to modifying BERT model such as adding PAL on top, adding PALs within the BERT model and use of different functions for the transformations within the PAL. In this project we implemented only one version which is adding the PAL layers within the BERT layer. While our implementation provided comparable performance, it would be interesting to explore the results with some additional experiments using different hyper parameters.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

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

[3] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2018.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.