BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning (2020)

Asa Cooper Stickland, Iain Murray

Summary

Contributions

In the simplest form of multi-task learning for various Natural Language Understanding (NLU) tasks, given a pre-trained language model, we can just fine-tune a new model for each given task. However, this way, for each task, we add 100% of the model's parameters to the parameters we need to store, making this approach infeasible for mobile devices and for application with a large number of tasks. To solve this problem, in this paper, the authors propose to use a small number of task-specific parameters in addition to the base parameters that are not modified during the fine-tuning approach, thus significantly reducing the number of parameters per task. To be more precise, the authors propose a new attention module, called a Projected Attention Layer or PAL, to be added in parallel to the normal BERT layers in addition to the appropriate training adjustments to avoid any catastrophic forgetting, and obtain competitive results compared to the standard fine-tuning approach but with 7x fewer parameters on the GLUE benchmark.

Method

Before starting the design process, the authors propose to limit the number of parameters to 1.13x of the original parameters, and given that BERT contains 110 million parameters, so the total number of parameters of all the tasks is upper limited by 125 million parameters, or 15 million additional parameters for all the tasks.

The BERT model consist of a series of multi-head attention BERT layers that transforms hidden states for each element of a sequence based on the other elements. Each attention layer consist of three matrices (key, query and value) of size $d/n \times d$ with d as the size of the hidden vector, and n as the number of head, in addition to the output matrix of size $d \times d$. Ignoring the bias terms, the total number of parameters of the attention layer is $3nd^2/n + d^2 = 4d^2$. Followed by an attention layer, we have layer normalization layer requiring $2d$ parameters, a fully connected layer with a inner dimension of d_{ff} requiring $2dd_{ff}$ and then an other layer normalization. So ignoring the linear number of parameters of the layer norm, in total, each BERT layer have $4d^2 + 2dd_{ff}$. Finally, the number of parameters of the BERT base model with 12 layers and a final pooling layers with $d \times d$ on top of the [CLS] token can easily be computed.

- Top Layer. Now, in order to reduce the number of parameters to be added per tasks, the authors proposed to first start by reducing the number of parameters required for the top layer, called TS or Task Specific layer. Instead of a single linear layer of $d \times d$ parameter, we have a task-specific functions of the form:

$$\mathrm{TS}(\mathbf{h}) = V^D g\left(V^E \mathbf{h}\right)$$

with an encoder V^E, a decoder V^D with an inner hidden size $d_s < d$, and an arbitrary function $g(.)$ with two possible choices: (1) Projected Attention: a Multi-head attention followed by a residual

connection and a layer norm with $d_s = 204$. (2) a layer ($d_s = 408$) or two ($d_s = 252$) of feed forward network followed by a residual connection and layer norm.

- Intermediate Layers. The second step is to define the per BERT layer parameters to be added. For this the authors proposed to add a similar TS (task specific) layer parallel to each self-attention layer (SA), in this case, for a given layer l, a hidden state $\mathbf{h}$, and a layer norm LN, the new output is:

$$\mathbf{h}^{l+1} = \text{LN}\left(\mathbf{h}^l + \text{SA}\left(\mathbf{h}^l\right) + \text{TS}\left(\mathbf{h}^l\right)\right)$$

with $\text{TS}(\mathbf{h}) = V^D g\left(V^E \mathbf{h}\right)$ similar to the top layer, but with different design choices for $g(.)$, as follows:

- The identity function with $d_s = 100$.
- Multi-head attention without the output matrix and with $d_s = 84$.
- Multi-head attention, with shared encoders and decoders across layers (1-12) with $d_s = 204$, this choice is refereed to as **ProjectedAttention Layers (PALs)**.
- Feed-forward and shared encoder and decoder across layers with $d_s = 204$.

For a depiction of this design see the figure bellow.

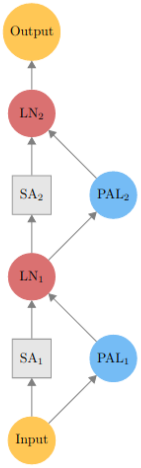

Figure 1. Schematic diagram of adding a task-specific function (here our 'Projected Attention Layers' or PALs) in parallel with self-attention (SA) layers in a BERT model (see section 3.3), with only two layers for simplicity. LN refers to layer-norm.

Training Details. Now that we defined the new parameters to add per tasks, the authors also propose an adjustment to the training setup. The normal setup consists of cycling through each task in a fixed order (round-robin), but given that each task have a different number of examples, such a sampling could lead to overfitting some tasks and/or forgetting other. To avoid this the authors propose to sample the tasks with a probability proportional to the number of examples per tasks N. Specifically, given that the difference between the bigger and smallest tasks might be big, resulting in an under-sampling of the small tasks, the sampling probability is $p_i \propto N_i^\alpha$ with $\alpha < 1$ to reduce the sampling disparity between the different tasks. Additionally, the authors noticed that it is better to go back to a normal equal sampling towards the end of training by annealing α as follows $\alpha = 1 - 0.8\frac{e-1}{E-1}$.

Results

Table 2. GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. We show F1/accuracy scores for QQP and MRPC, and accuracy on the matched/mismatched test sets for MNLI. The 'Av.' column is slightly different than the official GLUE score, since we exclude WNLI. 'Bert-base' results are from Devlin et al. (2018). 'Shared' refers to the model where all parameters are shared except the final projection to output space. The models we tested are a result of the 'annealed sampling' method for multi-task training as it produced the best results on the dev set.

METHOD	PARAMS	MNLI-(M/MM) 392K	QQP 363K	QNLI 108K	SST-2 67K	COLA 8.5K	STS-B 5.7K	MRPC 3.5K	RTE 2.5K	AV.
BERT-BASE	8×	84.6/83.4	89.2/71.2	90.1	93.5	52.1	85.8	84.8/88.9	66.4	79.6
SHARED	1.00×	84.0/83.4	88.9/70.8	89.3	93.4	51.2	83.6	81.3/86.7	76.6	79.9
TOP PROJ. ATTN.	1.10×	84.0/83.2	88.8/71.2	89.7	93.2	47.1	85.3	83.1/87.5	75.5	79.6
PALS (204)	1.13×	84.3/83.5	89.2/71.5	90.0	92.6	51.2	85.8	84.6/88.7	76.0	**80.4**

Table 3. GLUE performance, in terms of average score across each task's development set; this score is accuracy except for CoLA, where it is Matthews correlation, and STS-B, where it is Pearson correlation. We show the mean and standard error over three random seeds, unless standard error is < 0.005. For the details of the sampling strategies see section 4.1. For the 'within BERT' methods we show the smaller hidden state size in brackets, and write 'no sharing' to refer to not sharing V^E and V^D across layers, 'top' to mean adding in parallel to the six BERT layers just before the output, and 'bottom' to mean adding in parallel to the six BERT layers just after the input.

METHOD	NO. PARAMS	NEW LAYERS	PROP. SAMP.	SQRT. SAMP.	ANNEAL SAMP.
SHARED	1.00×	0	79.17±0.03	80.56±0.04	80.7±0.3
ADDING ON TOP OF BERT					
BERT LAYER	1.66×	1	80.6±0.2	81.6±0.3	81.5±0.2
PROJ. ATTN.	1.10×	6	80.3±0.1	81.4±0.1	81.5±0.1
PROJ. FFN (1 LAYER)	1.10×	6		81.07	80.8±0.1
ADDING WITHIN BERT					
PALS (204)	1.13×	12	80.6±0.2	81.0±0.2	**81.7±0.2**
PALS NO SHARING (84)	1.13×	12			81.3±0.1
LOW RANK (100)	1.13×	12			**81.9±0.2**
PALS (276, TOP)	1.13×	6			81.61±0.06
PALS (276, BOTTOM)	1.13×	6			81.4±0.1