BERT-A: Fine-tuning BERT with Adapters and Data Augmentation

Introduction and Problem

- In recent years, two trends in NLP research:
  - Pre-trained contextual embeddings: ELMo, BERT, etc.
  - Multi-task learning: Decathlon, GLUE, etc.

How can we use both?

A naive approach would result in millions of additional parameters per task. These need to be stored and loaded for each inference.

Problem: Question Answering

Dataset: SQuAD 2.0: (paragraph, question) pairs, either the answer is a span in paragraph or there are no answers.

Goals:
- Improve performance in terms of F1 and exact match (EM) scores
- Keep additional trainable parameters to a minimum

Storage Efficient Results

Here we aim to maintain performance while minimizing number of additional parameters.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>EM</th>
<th># Parameters (Overhead)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline (fine-tuned)</td>
<td>76.5</td>
<td>73.5</td>
<td>110 M (+100%)</td>
</tr>
<tr>
<td>baseline (top block fine-tuned)</td>
<td>54.0</td>
<td>51.7</td>
<td>9.2 M (+83.6%)</td>
</tr>
<tr>
<td>baseline (frozen)</td>
<td>51.1</td>
<td>51.0</td>
<td>1.5 K (+0.001%)</td>
</tr>
<tr>
<td>baseline (frozen) + PALS (120)</td>
<td>63.9</td>
<td>60.7</td>
<td>704 K (+0.64%)</td>
</tr>
<tr>
<td>baseline (frozen) + Adapters</td>
<td>70.9</td>
<td>67.4</td>
<td>592.9 K (+0.54%)</td>
</tr>
<tr>
<td>baseline (frozen) + Adapters</td>
<td>74.7</td>
<td>72.3</td>
<td>629.7 K (+0.57%)</td>
</tr>
</tbody>
</table>

Adapters consistently outperform other approaches in QA. We achieve comparable performance with just 0.57% additional parameters to store per task.

Approach

- We experiment with two types of task-specific modules inside each Transformer block
  - Adapters
  - Projection Attention Layers
- We use an Answer Pointer layer as our output layer
- For better performance, we use:
  - data augmentation (increase instances of "no-answer" questions)
  - transfer learning from CoQA dataset

Performance Results

Here we sacrifice storage efficiency for performance.

We train on SQuAD 2.0, CoQA, no-answer augmented datasets Adapters are trained after other parameters are done training.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1 (dev)</th>
<th>EM (dev)</th>
<th>Training time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline (fine-tuned)</td>
<td>76.5</td>
<td>73.5</td>
<td>377</td>
</tr>
<tr>
<td>baseline + Answer Pointer</td>
<td>76.7</td>
<td>73.5</td>
<td>380</td>
</tr>
<tr>
<td>baseline + Data Augmentation</td>
<td>77.9</td>
<td>75.5</td>
<td>1110</td>
</tr>
<tr>
<td>baseline + Pre-training on CoQA</td>
<td>78.5</td>
<td>75.7</td>
<td>836</td>
</tr>
<tr>
<td>baseline + Pre-training on CoQA + Adapter</td>
<td>79.2</td>
<td>76.3</td>
<td>1240</td>
</tr>
<tr>
<td>baseline + Pre-training on CoQA + Data Augmentation + Answer Pointer</td>
<td>79.5</td>
<td>76.5</td>
<td>1722</td>
</tr>
<tr>
<td>baseline + Pre-training on CoQA + Data Augmentation + Answer Pointer + Adapter</td>
<td>80.5</td>
<td>77.5</td>
<td>2151</td>
</tr>
</tbody>
</table>

Test set scores: F1: 81.44 (3rd) EM: 78.36

Analysis

- Adapters’ weights in the last transformer block:
  - We probably did not need adapters in self-attention.
  - But adapters in all blocks in output learned patterns.

- Performance improvements for longer distribution for non-trivial questions.

Conclusion and Future Work

- Using Adapters with frozen BERT is an effective way to decrease per task parameters in a multi-task learning setting.
- Fine-tuning BERT with Adapters can increase the performance in terms of F1 and EM scores without overfitting.
- Even simple data augmentation techniques work well compared to architectural changes after the top layer of BERT.
- Future work: Assessing interpretability of task-specific modules inside BERT

References