**Introduction**

Question Answering is a task in Natural Language Processing where an algorithm attempts to answer a question given context that would help answer it. For this project, ALBERT, a BERT-based neural network was evaluated on the SQuAD question answering dataset. Some hyperparameters were tuned such as learning rate and batch size to gain a better understanding of what affects model performance. Through this tuning, we achieve a F1 Score of 90.37, compared to the published results of an F1 of 89.3.

**Process**

The HuggingFace Transformers library was used to download a pre-trained version of ALBERT, a version of BERT that reduces memory footprint and increases training speed by reducing the number of model parameters needed. Additionally, the SQuAD dataset was used to finetune against, also retrieved from HuggingFace. Finally, all the experiments were tracked using the Weights and Biases tool. A total of 5 different batch sizes and 4 different learning rates were evaluated. Each model was finetuned for a total of 3 epochs and then evaluated, regardless of how close to convergence the training process was. A weight decay of 0.01 was used on all runs, as suggested by the original HuggingFace tutorial. All experiments were run on a NVIDIA RTX 2080 TI with 11 GB of VRAM.

**Results**

The results shown below show the effect of batch size and learning rate on the F1 performance of each fine-tuning run. Additionally, runtime information was collected to assess the impact of the different hyperparameters on the training time. Finally, memory usage information was collected to asses how the hyperparameters varied memory utilization.

|  |  |
| --- | --- |
|  |  |
|  |  |

**Charts that show the effect of learning rate and batch size on F1 score.**

|  |  |
| --- | --- |
|  |  |

**Training Runtime and Memory Usage by Batch Size. Learning Rate had no discernable difference in runtime and memory usage.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Batch Size 1** | **Batch Size 2** | **Batch Size 4** | **Batch Size 8** | **Batch Size 12** |
| **LR 0.000002** | N/A | 89.928 | N/A | N/A | 89.382 |
| **LR 0.00002** | 87.294 | 88.717 | 89.288 | 90.138 | **90.37** |
| **LR 0.0005** | N/A | N/A | N/A | N/A | 88.978 |
| **LR 0.002** | N/A | N/A | N/A | N/A | 3.421 |

**Complete Results of Fine-tuning with varying Hyperparameters. Not all combinations were evaluated due to time constraints.**

|  |  |
| --- | --- |
| **Method** | **F1 Score** |
| ALBERT (Paper) | 89.3 |
| **{ANNA} (SOTA)** | **95.72** |
| ALBERT (Ours) | 90.37 |

**Comparison of the fine-tuned model and other notable models.**

**Discussion**

The results of the hyperparameter tuning can be seen above. Notably, the larger batch sizes performed better than smaller batch sizes. This is probably due to the fact that larger batch sizes promote gradient updates that are more beneficial to the entire dataset rather than a specific sample or few examples. Additionally, learning rate is a critical parameter. Too large of a learning rate (such as 0.002 as evaluated), and the optimization process never converges. In this case, it didn’t even start to find an optimal solution. A too small learning rate also hampers the process, as the model does not find as optimal of a solution in the same number of training steps, due to the fact the steps it takes do not follow the gradient aggressively enough.

The training runtime and memory usage results are also interesting. We can see that there is not a significant increase in memory usage between a batch size 1 and a batch size of 2, suggesting that a large portion of the 20% memory utilization with a batch size of 1 is simply for storing the model in the GPU’s memory. Unfortunately, larger batch sizes greater than 12 were not evaluated due to the memory constraints on the GPU, although the trend suggests that an even larger batch size would further improve the results, although probably with considerable diminishing returns. Another benefit of larger batch sizes, however, is not just an improvement in F1, but also a decrease in training time, as fewer gradient updates have to be calculated for the same number of epochs.

Finally, when comparing the result of the fine-tuning and hyperparameter search performed here to the original ALBERT paper and ANNA, the current leader of the SQuAD task according to Papers with Code, we see that we have a slight edge over the original ALBERT paper. This could be a legitimate lift, or just random luck. Further runs over the same set of hyperparameters would allow us to determine if there is a benefit to our specific set of hyperparameters compared to the original training regime.

**Conclusion**

Through a hyperparamter tuning process, a F1 score of 90.37 was achieved fine-tuning a pre-trained ALBERT model on the SQuAD dataset. This process showed the importance of both the learning rate and batch size hyperparameters. It was determined that the best combination of hyperparameters tested is a learning rate of 0.00002, with a batch size of 12. Some important trends observed were that a larger batch size both increases F1 score, but also decreasing the training time to fine-tune the model. Additionally, a good learning rate is finding a balance of one that is large enough to take meaningful steps towards the minima during the training process, but one that is small enough so that the optimization process eventually converges.

**References**

The code used is from this HuggingFace tutorial notebook: <https://colab.research.google.com/github/huggingface/notebooks/blob/master/examples/question_answering.ipynb>

Some light modifications to use Weights & Biases were found here:

<https://docs.wandb.ai/guides/integrations/huggingface>

Papers with Code SQuAD Leaderboard:

<https://paperswithcode.com/sota/question-answering-on-squad11?metric=F1>

ALBERT Paper:

@article{DBLP:journals/corr/abs-1909-11942,

author = {Zhenzhong Lan and

Mingda Chen and

Sebastian Goodman and

Kevin Gimpel and

Piyush Sharma and

Radu Soricut},

title = {{ALBERT:} {A} Lite {BERT} for Self-supervised Learning of Language

Representations},

journal = {CoRR},

volume = {abs/1909.11942},

year = {2019},

url = {http://arxiv.org/abs/1909.11942},

eprinttype = {arXiv},

eprint = {1909.11942},

timestamp = {Fri, 27 Sep 2019 13:04:21 +0200},

biburl = {https://dblp.org/rec/journals/corr/abs-1909-11942.bib},

bibsource = {dblp computer science bibliography, https://dblp.org}

}

SQuAD Paper:

@article{DBLP:journals/corr/RajpurkarZLL16,

author = {Pranav Rajpurkar and

Jian Zhang and

Konstantin Lopyrev and

Percy Liang},

title = {SQuAD: 100, 000+ Questions for Machine Comprehension of Text},

journal = {CoRR},

volume = {abs/1606.05250},

year = {2016},

url = {http://arxiv.org/abs/1606.05250},

eprinttype = {arXiv},

eprint = {1606.05250},

timestamp = {Mon, 24 Aug 2020 14:01:25 +0200},

biburl = {https://dblp.org/rec/journals/corr/RajpurkarZLL16.bib},

bibsource = {dblp computer science bibliography, https://dblp.org}

}

Code can be accessed on GitHub: https://github.com/parkererickson/csci8980-hw2