Optimising for Smaller Model and Faster Training on Question-Answering Bert Models in a Constrained Environment

A0196717N, A0205773A, A0219763W, A0268425X, A0236178X
Group 10
Mentored by Hai Ye
{e0389203,e0425696,e0550421,e1101724,e0732728}@u.nus.edu

Abstract

While BERT has proven itself to be an effective architecture to tackle the Stanford Question Answering Dataset (SQuAD), fine-tuning the model often takes a significant amount of time which prevents NLP practitioners who do not have access to powerful GPUs to experiment with it. Moreover, its massive size might not be viable for many applications such as mobile applications. In this report, we experimented with ways to maximise DistilBERT's (a smaller variant of BERT) performance while reducing its size under very specific conditions: less than 20 minutes of training with GPU T4(x2) using only 10,000 data points. Under this setting, we concluded that adding certain CNN or LSTM-based layers after DistilBERT's last hidden layer led to marginal improvement while fine-tuning DistilBERT with response-based Knowledge Distillation yielded a moderate improvement compared to the conventional fine-tuning procedures under Exact Match (EM) and F1 metrics. Furthermore, some magnitude-based weight pruning can reduce DistilBERT's size by promoting sparsity while maintaining baseline performance.

1 Introduction

Question-Answering (QA) is one of the major fields of Natural Language Processing (NLP), where the task is for machines to be able to understand and answer questions posed in the natural language. The Stanford Question Answering Dataset (SQuAD) is one of the benchmark datasets for evaluating QA systems. It is a reading comprehension dataset composed of questions created by crowd workers on a set of paragraphs extracted from top Wikipedia articles, where the answer to the question is strictly a text segment of the original paragraph (Rajpurkar et al., 2016). This paper will use the SQuAD v1.1 dataset.

Bidirectional Encoder Representations from Transformers (BERT) is a family of masked-language models that was introduced by Google in 2018. In just a few years, it has proven itself to be a highly effective model in many NLP tasks, including Question-Answering (Devlin et al., 2018). A popular method to contextualise BERT to a specific problem is fine-tuning, where after BERT is pre-trained on language modelling and next sentence prediction, it is further trained on a smaller dataset for a specific task. Even though the fine-tuning process is significantly less computationally expensive than the pre-training process, BERT can still take a long time to fine-tune (hours to run one epoch of training on SQuAD v1.1 on GPU T4(x2)), not to mention its clunkiness due to its hundred million parameters. This is unsustainable as many NLP practitioners do not have access to powerful GPUs and many applications (e.g. mobile applications) might not be able to support a model size as big as BERT.

Throughout the years, variants of BERT have been developed to reduce its training time and size, including ALBERT (Lan et al., 2020) and DistilBERT (Sanh et al., 2020). However, even fine-tuning these variants for one epoch on the whole SQuAD v1.1 dataset still takes more than 1 hour on GPU T4(x2).

In this report, we will use DistilBERT and explore ways to optimise its performance on SQuAD v1.1 according to standard evaluation metrics (Exact Match and F1 score) with limited training time (<= 20 minutes of fine-tuning on GPU T4(x2)) and limited data (10,000 training data points) and discuss our findings. Furthermore, we will also explore ways to reduce DistilBERT's size without compromising much of performance[1].

Through our experiments and analysis, we conclude that using knowledge distillation with BERT large as the teacher model for a DistilBERT student model yielded the best result for limited training

[1]DistilBERT was chosen instead of other models like ALBERT as our experiment concluded that it was the fastest to train.

time and limited data, while adding certain CNN or LSTM-based layers after DistilBERT's last hidden layer led to some improvement. Meanwhile, some magnitude-based weight pruning can reduce DistilBERT's size without compromising performance discernably.

2 Related Work / Background

Since SQuAD was released, there have been many attempts to utilise this dataset for the purpose of the question answering model generation. The most common metrics used to gauge the performance of a QA model are Exact Match (EM) and macro-averaged F1 score (Rajpurkar et al., 2016).

EM refers to the percentage of the predictions that match the ground truth answers exactly. Meanwhile, F1 score measures the overlap between the predictions and the ground truth answers. Both the predictions and the ground truth answers are treated as bags of token, and their F1 is computed. The macro-averaged F1 score refers to the average of all the questions' F1 scores. In our report, we will also use these 2 metrics.

While early works in QA typically involved a rule-based model with feature engineering (Utah et al., 2000), most modern approaches to tackling SQuAD involve a deep-learning architecture such as BERT that predicts the index of the tokens where the answer starts and ends separately. One of the earliest examples to use this approach is Stanford Attentive Reader, a biLSTM-based architecture which was developed in 2017 (Chen et al., 2017).

To contextualise BERT-based models to QA, an additional linear output layer is typically used to convert the tokens' vectors to prediction logits for both the start and end index. This is the method used on SQuAD by Google AI Language team when they first introduced BERT (Devlin et al., 2018). However, adding layers other than a linear one have been shown to occasionally improve F1 score (Zhang et al., 2019). This approach has not been tested with limited training time and data points and thus we would try this approach and assess its efficacy. In our experiments, we will try several Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) based architectures.

Knowledge Distillation (KD) is a method to transfer knowledge from a larger and more complex model – commonly referred to as the teacher – to a smaller and simpler model – commonly referred to as the student (Hinton et al., 2015). It has been proven to work wonderfully to distil BERT's massive knowledge. In fact, DistilBERT itself is a model that is pre-trained using BERT as its teacher (Sanh et al., 2020). In our experiment, we will use Knowledge Distillation using pretrained DistilBERT as the student, and finetuned BERT Large as the teacher. As typically Knowledge Distillation will lead to faster convergence (He et al., 2022), we hypothesise that this approach will work well for limited training time.

Lastly, to attempt to reduce the size of DistilBERT, we tried magnitude-based weight pruning on the model. Pruning refers to a method to achieve the goal of reducing the model size of a deep learning model while preserving performance while magnitude-based weight pruning aims to zero out model weights to achieve sparsity, which will lead to better compression. While multi-head attention, a major driving force behind models like BERT, tends to perform better as each attention head potentially focuses on different parts of the input, it is also seen that a large percentage of these heads can be removed without impacting performance (Michel et al., 2019). As pruning has been proven to work well on BERT-like models to reduce their sizes without impacting performance (Sajjad et al., 2022), we tried this approach on the DistilBERT model with the before-mentioned constraints.

3 Corpus Analysis & Method

We chose Huggingface's pretrained DistilBERT as our baseline question answering model due to its versatility and extensive documentation. The full code of this report can be found on our group's repository[2].

3.1 Corpus Analysis

1. **Data Collection** SQuAD v1.1 dataset consists of paragraphs sampled randomly from top 10,000 articles on Wikipedia, with questions and answers supplied by crowd workers.

2. **Preprocessing** We followed Huggingface's standard preprocessing steps (Figure 1) for QA models (Huggingface.co, 2023)[3], which are as follows:

[2] https://github.com/bernarduskrishna/CS4248-Project

[3] https://huggingface.co/docs/transformers/tasks/question_answering

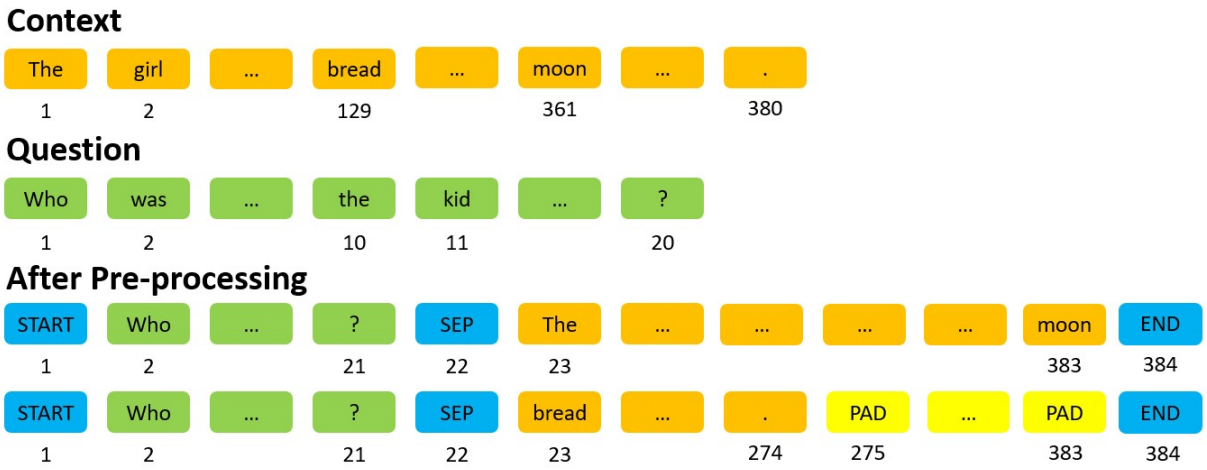

Figure 1: SQuAD preprocessing

- We first tokenise all questions and the contexts using Huggingface's Tokeniser library. Depending on which version of BERT/its variant one uses (cased or uncased), case normalisation might be done.
- For each question-answer pair, we will combine them into one list with a special separator token in between them. However, this method might lead to variable list sizes, and thus we limit the list size to 384. If the context is too long, it will be split into several lists with stride size of 128. Any excess space will be filled with a special padding token.
- We will also store the attention masks that indicate where the paddings are.
- The index of the starting position of the answer as well as the ending position of the answer will also be stored.

3.2 Main methods used

3.2.1 Knowledge Distillation

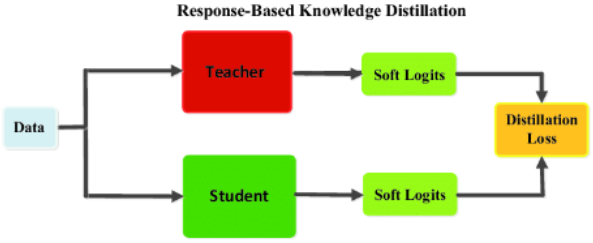

Figure 2: Response-based Knowledge Distillation

In our experiment we will be using response-based knowledge distillation (Gou et al., 2021), where the student directly mimic the prediction of the teacher model. In other words, given the teacher's soft-maxed prediction logits q_t, and the student's prediction logits z_s, the loss function will be computed as $L_{\text{KD}}(q_t, z_s)$, where L_{KD} is the cross-entropy loss function.

Our experiments will involve:

1. **Changing Loss Function** While the original paper uses cross-entropy as its loss function, some papers have suggested using Kullback-Leibler Divergence as loss function (Cho and Hariharan, 2019). Thus, we will be comparing the performance between the two under our experimental setting.

2. **Changing Temperature** Temperature refers to the scaling factor that is applied to the prediction logits before softmax function is applied (Tang et al., 2021). The softmax function is defined as follows:
$(\text{q}_i) = \text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}$
Upon the introduction of temperature, we will then scale the prediction logits before the softmax function is applied.
$(\text{q}_i\text{'}) = \text{softmax}(z_i/T) = \frac{\exp(z_i/T)}{\sum_{j=1}^{K} \exp(z_j/T)}$
We will experiment with the temperature values of 0.5, 1, and 2.

3. **Changing α** When the actual correct labels are known, it is possible to use the weighted average between two loss functions (Hinton et al., 2015). The first loss function will be the L_{KD} described above, while the second

loss function will be the cross entropy with the correct labels, L_S. The total loss function is then defined as: $\alpha L_{\text{KD}} + (1 - \alpha)L_S$.

We will try α values of 0.5, 0.7, and 0.9.

3.2.2 Adding additional layers on top of the DistilBERT model

In our experiment, we swap out the output layer of the DistilBERT model and append additional layers on top of it. The additional layers take the output of DistilBERT model's final hidden layer as input and output the predicted start logit and end logit. We load the DistilBERT model with its pretrained weights and randomly initialized the weights of the additional layers, then fine-tune the whole model end-to-end. Notice that since the hidden size of the DistilBERT model is 768, all the additional layers will use this as the input size.

Our experiments will involve:

1. **Adding LSTM-based Layers** Long Short-Term Memory (LSTM) is a RNN-based architecture proposed by (Hochreiter and Schmidhuber, 1997) which adds explicit sequential information into the model and may help improving the results (Zhang et al., 2019). We pass the output of the LSTM to a fully-connected layer that has a output dimension of 2 after each time step to compute the start and end logits. The following LSTM configurations will be experimented:

    - LSTM 1: DistilBERT + LSTM(1 layer, 768 hidden, bidirectional) + FC(1536 in, 2 out)
    - LSTM 2: DistilBERT + LSTM(1 layer, 64 hidden, bidirectional) + FC(128 in, 2 out)
    - LSTM 3: DistilBERT + LSTM(2 layers, 64 hidden, bidirectional) + FC(128 in, 2 out)

2. **Adding CNN Layers** While RNN-based layers can model long-term dependencies, Convolutional Neural Network (CNN) layers can be used after BERT-based models to capture local relations of the sequence and filter out salient features (Murfi et al., 2022).

    We will convolve the output of the DistilBERT with 2D convolution. Since the output of the DistilBERT model is in the shape of (batch_size, sequence_length, hidden_size), but the CNN layer would expect an input shape of (batch_size, num_channel, height, width), we treat the DistilBERT output as a single channel input and unsqueeze it to the shape of (batch_size, 1, sequence_length, hidden_size). Convolving along the hidden state dimension is generally not explicable as the elements of the each hidden state is assumed to be not correlated. Therefore, we only convolve along the sequence dimension and use a kernel shape of (n, hidden_size). By padding the sequence dimension, the convolution will produce an output of (batch_size, num_channel, sequence_length, 1) shape. We use ReLU as the activation function, followed by a convolution layer with 2 output channels and 1x1 kernel to output the prediction logits.

    The following CNN configurations will be experimented:

    - CNN 1: DistilBERT + CNN (Cin=1 Cout=24 kernel=(3, hidden_size) stride=1 pad=(1,0)) + ReLU + CNN (Cin=24 Cout=2 kernel=1 stride=1 pad=0)
    - CNN 2: DistilBERT + CNN (Cin=1 Cout=64 kernel=(3, hidden_size) stride=1 pad=(1,0)) + ReLU + CNN (Cin=64 Cout=2 kernel=1 stride=1 pad=0)
    - CNN 3: DistilBERT + 3 CNN (Cin=1 Cout=8 kernel=((3 and 5 and 7), hidden_size) stride=1 pad=(1,0) and (2,0) and (3,0)) + ReLU + concat + CNN (Cin=24 Cout=2 kernel=1 stride=1 pad=0)

    For CNN 3, we convolve the DistilBERT output with 3 convolution layers using different kernel sizes of (3, hidden_size), (5, hidden_size), and (7, hidden_size) correspondingly, with padding size (1, 0), (2, 0), and (3, 0) respectively. The results are concatenated along the channels dimension.

3.2.3 Pruning layers on the DistilBERT model

In our experiment, we will use magnitude-based weight pruning to zero out nodes that have less than a certain threshold on their weight values. This is performed by adjusting the pruning ratio (PR) as not all parameters are strictly needed and most of them can be removed or reallocated without

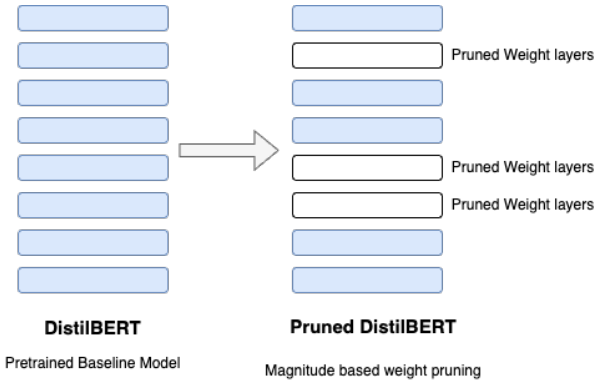

Figure 3: Magnitude based weight pruning

significantly impacting performance (Gordon et al., 2020).

We utilised magnitude-based weight pruning on a pre-trained DistilBERT model by gradually reducing its sparsity from 0% to 50% in increments of 10%. This method involves two steps, as described in the paper: first, the weight information is removed by setting it to 0, and then the model is finetuned on the new datasets during training. To implement this, we set the weights to 0 effectively deleting the pre-training information associated with the weight. Thereafter, during downstream training the model was allowed to fit into the new datasets.

4 Experiments

Corpus used We will be using the SQuAD v1.1 dataset. However, as we seek to explore ways to optimise DistilBERT's performance under limited training time and limited data, we will only be using the first 10,000 question-answer pairs from the train dataset. Meanwhile, we will be using the first 1,000 data points from the SQuAD v1.1 validation dataset as our validation set.

Baseline Our baseline will be DistilBERT with 6 layers and 66M parameters with case normalisation (distilbert-base-uncased on Huggingface's pretrained model list). The hyperparameters chosen are learning rate of 2E-5, an exponential scheduler with gamma 0.9, and a batch size of 4. The justification of these hyperparameters can be found at the Appendix.

This baseline model reached **EM of 53.7%** and **F1 score of 0.683**.

Experimental Results

4.1 Knowledge distillation (KD) with BERT large as teacher model

Model	EM	F1
Baseline	53.7%	0.683
KD (Loss: CE, α = 1, T = 0.5)	**57.5%**	**0.714**

Table 1: Best model in Knowledge Distillation experiment

Referring to Table 1, using the best Knowledge Distillation settings according to our experiments, EM increased by 3.8% and F1 score increased by 0.031 compared to the baseline.

- Changing loss function to Kullback-Leibler Divergence

  Using Cross Entropy as loss function yields a much better result.

Model	EM	F1
KD (Loss: CE)	**57.3%**	**0.712**
KD (Loss: KL Divergence)	53.5%	0.661

Table 2: KL Divergence loss Experiment

- Changing the temperature

  A lower temperature generally results in better EM and F1.

Model	EM	F1
KD(T = 0.5)	**57.5%**	**0.714**
KD(T = 1)	57.3%	0.712
KD(T = 2)	55.5%	0.699

Table 3: Changing temperature experiment

- Changing α (mixing ratio between distillation loss and actual loss)

  A pure KD approach (without any weight given to the loss from the true labels) seems to perform better. As the weight given to the loss from true labels (1 - α) increases, results tend to deteriorate.

Model	EM	F1
KD(α = 0.5)	54.5%	0.682
KD(α = 0.7)	55.6%	0.688
KD(α = 0.9)	56.2%	0.701
KD(α = 1)	**57.3%**	**0.712**

Table 4: Changing α experiment

4.2 Adding additional layers on top of the DistilBERT model

Two out of the six experiments produced a better F1 score when compared to the baseline. The best LSTM-based add-on improved the F1 score by 0.005 and the best CNN-based add-on improved the EM by 2.0%.

Model	EM	F1
Baseline	53.7%	0.683
LSTM 2	55%	**0.688**
CNN 3	**55.7%**	0.684

Table 5: Best model in adding layers experiment

- Adding LSTM-based layers

  Adding a small LSTM layer can marginally improve the result, but a larger LSTM layer hurts the performance.

Model	EM	F1
LSTM 1	52.7%	0.679
LSTM 2	**55.0%**	**0.688**
LSTM 3	53.4%	0.678

Table 6: Adding LSTM layers experiment

- Adding CNN-based layers

  Using kernel size of 3 along the sequence dimension generally hurts the performance, but combining the kernel sizes of 3, 5, and 7 produces slightly better result.

Model	EM	F1
CNN 1	52.6%	0.668
CNN 2	54.2%	0.679
CNN 3	**55.7%**	**0.684**

Table 7: Adding CNN layers experiment

4.3 Magnitude based weight pruning

Using the pruning method's default settings and tuning the pruning threshold ratio on the pretrained DistilBERT, EM increased by 0.3% and stayed within the same range of F1 when the magnitude weight based pruning had a threshold of 0.3, which is 30%. This pruned model with reduced size has non significant changes on the accuracy over the baseline model.

Model	EM	F1
Baseline	53.7%	0.683
Pruning threshold: 0.3	**54%**	0.68

Table 8: Best model in pruning experiment

- Changing Pruning magnitude threshold

  Increasing the pruning ratio further impacts the performance metrics of the model as visible through the results.

Model	EM	F1
Pruning threshold: 0.1	51.5%	0.673
Pruning threshold: 0.2	52.9%	0.678
Pruning threshold: 0.3	**54%**	**0.68**
Pruning threshold: 0.4	50.7%	0.656
Pruning threshold: 0.5	50.5%	0.653

Table 9: Changing pruning ratio threshold experiment

5 Discussions

5.1 Knowledge Distillation with BERT large as teacher model

Knowledge distillation works very well for our experiment setting as we use a small subset of the whole SQuAD dataset and we have limited training time. The teacher model has already seen the whole SQuAD dataset and thus its prediction logits will already include the knowledge of the rest of the dataset that the student model does not see. The teacher's prediction logits contain much more information that the simple one-hot encoded answers, and thus the student can learn more from each iteration, resulting in faster convergence time.

Usually, a higher temperature is recommended for models of similar size (e.g. for self-distillation)

as a higher temperature will cause the soft-labels distribution of the teacher's prediction logits to be richer in information (Hinton et al., 2015). However, when the student's model is much smaller than the teacher's (as in the case for DistilBERT and BERT Large), the student's model might not have the capacity to learn that much information. Thus a smaller temperature should yield a better result, as shown by our experiment.

Although other papers showed that allocating some weight to the loss from the true labels (i.e. $\alpha < 1$) should yield better results, we could not replicate it in our experiment. We hypothesise that training time of 20 minutes with GPU T4(x2) is not enough for DistilBERT to learn the extra information provided by the loss from the true labels.

5.2 Adding additional layers on top of the DistilBERT model

The experiments with additional layers could only yield marginal improvements on the result, which is roughly consistent with the prior experiments conducted by (Zhang et al., 2019). One explanation can be that the DistilBERT model is already capable of performing what the additional layers are able to. While LSTM layers can model the temporal dependences, DistilBERT may also be able to do so with the help of the position embeddings. While CNN can filter local features, DistilBERT may be able to capture both local and global features with the self attention mechanism. Nevertheless, as the weights of the additional layers are randomly initialized, it may also be the case that the models with result improvements are initialized with "good" weights by chance, given that the improvement is not very salient.

While adding a single layer bidirectional LSTM with a hidden size of 64 appear to improve the model's performance, both the experiment with more LSTM layers and with larger hidden size negatively affected the results. This may be the result of overfitting, as indicated by a strictly larger model size and a larger gap between the train accuracy and the validation accuracy.

For the CNN-based additional layer, all experiments with the kernel size of (3, hidden_size) failed to improve the model's results, while combining the results of kernel sizes of 3, 5, and 7 along the sequence dimension yielded a better result. This may suggest that smaller kernels may be insufficient to capture the local relationship of the sequence, whereas a combination of kernel sizes can better integrate the local context at different levels to aid the prediction.

5.3 Magnitude based weight pruning

In our constrained environment, we managed to reduce the size of the model by promoting sparsity without compromising performance, as can be seen as the comparable performance with the baseline.

In our experiment, it could be seen that the efficiency of the model decreased on lower thresholds when it was around 10% and 20%. One reason that could be attached to it is the constrained environment of our experiment which expects the model to learn in a limited time. The model's reduction in size (i.e. increased sparsity) is not sufficient for the model to learn fast enough during the fine-tuning step (Sajjad et al., 2022) within constrained time as accuracy improves over training time and hence fails to perform well compared to baseline model.

The experiment performed its best at the rate of 30% pruning at which state the learning was fast enough given the model's sparsity/size on the constrained time parameters. This model performed the closest to baseline in terms of F1 performance metrics and slightly performed better on the Exact Match (EM) scores.

As the pruning threshold ratio was increased further to 40% and 50%, the model's performance started to deteriorate. This results agree with (Gordon et al., 2020) in which the higher thresholds of pruning a pre-trained model do not perform well in fine-tuning on the downstream datasets (SQuAD dataset, in our case) due to the extreme loss of information when pruning was done.

5.4 General discussion on our model's performance

We took our best-performing model (Knowledge Distillation with temperature 0.5) and did a study on what kind of questions it did poorly.

As can be seen from figure 4, our model tends to perform badly in answering "where" and "why" questions compared to other types of questions. We further analysed why this could be the case. Upon further investigation, we concluded that it might be due to lack of training on these questions. Figure 5 shows the proportion of question types on the training set.

From figure 5, "where" and "why" questions constitute less than 5% of the training set, which

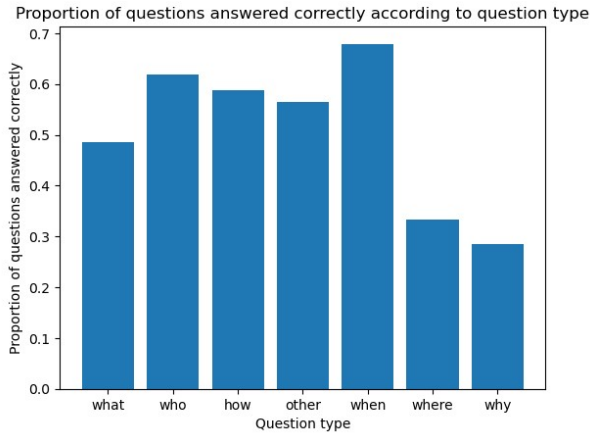

Figure 4: Proportion of questions answered correctly according to question type

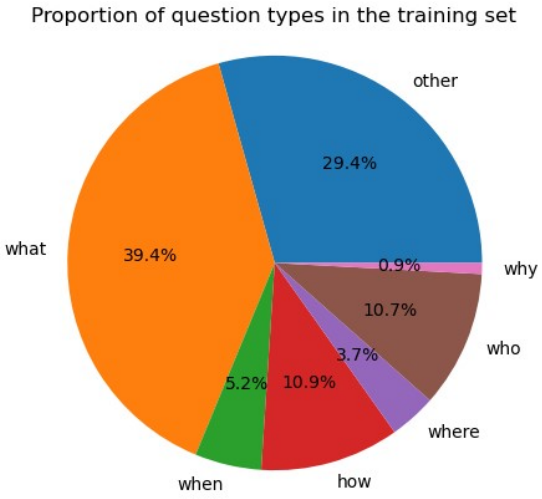

Figure 5: Proportion of question types in training set

may be the reason why our model cannot answer such questions satisfactorily.

6 Conclusion

Even with the recent push for lighter BERT-based models that require less training time, it is still untenably time-consuming even just to fine-tune BERT for a specific dataset. Our work shows that it is possible to achieve commendable performance with limited training time and limited dataset by experimenting with CNN and LSTM-based layers after DistilBERT's last hidden layers and utilising the idea of Knowledge Distillation. Furthermore, for such a constrained environment, we have shown that some pruning can reduce model size without compromising much of performance.

While we have done these three approaches in isolation due to time constraints, it remains to be explored how these three approaches will interact with one another. Furthermore, more rigorous experiments with many repeated tries for a specific setting to get an average and standard deviation will definitely lead to more robust results, although it will be extremely time-consuming.

Lastly, our analyses have shown that certain types of questions are underrepresented in the training set, such as "why" and "where" questions, leading to poor performance for such questions. Thus, it might be a sensible idea to balance the type of questions in future works whenever possible.

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Jang Hyun Cho and Bharath Hariharan. 2019. On the efficacy of knowledge distillation. *arXiv preprint arXiv:1910.01348*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mitchell A. Gordon, Zhengyuan Zhu, Arnav Gupta, Adarsh Suggala, Vaibhav Raghavan, Sanya Gupta, Rishabh Jha, Udbhav Khandelwal, and Srinivas Turaga. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*.

Jianping Gou, Zhaoyang Zhu, Jiaxin Fan, Wangmeng Zuo, and Lei Zhang. 2021. Knowledge distillation: A survey. *arXiv preprint arXiv:2006.05525*.

Ruifei He, Haichao Zhang, Fei Sun, Jian Gao, and Xiaodong Li. 2022. Knowledge distillation as efficient pre-training: Faster convergence, higher data-efficiency, and better transferability. *arXiv preprint arXiv:2203.05180*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *arXiv preprint arXiv:1905.10650*.

Hendri Murfi, Handi Tjandrasa, Ayu Purwarianti, and Herlina Andriani. 2022. Bert-based combination of convolutional and recurrent neural network for indonesian sentiment analysis. *arXiv preprint arXiv:2211.05273*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Hassan Sajjad, Francisco Guzman, Alan W Black, and Rafael E Banchs. 2022. On the effect of dropping layers of pre-trained transformer models. *arXiv preprint arXiv:2004.03844*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Jiaxi Tang, Liangzhen Deng, Junjie Yan, and Xiaolin Li. 2021. Understanding and improving knowledge distillation. *arXiv preprint arXiv:2002.03532*.

Ellen Riloff University of Utah, Pamela Jordan, Richard Katz, and Carl Sable. 2000. A rule-based question answering system for reading comprehension tests. In *Proceedings of the 2000 ANLP/NAACL Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems - Volume 6*.

Xu Zhang, Daya Han, Zhiyuan Liu, Maosong Sun, and Leyu Zhou. 2019. Bert for question answering on squad 2.0. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4238–4243.

Acknowledgements

Thanks to Hai Ye for his guidance throughout the project. Thanks to Prof Kan and Prof Weth for all the hard work during the entirety of this module.

Statement of Independent Work

1A. Declaration of Original Work. By entering our Student IDs below, we certify that we completed our assignment independently of all others (except where sanctioned during in-class sessions), obeying the class policy outlined in the introductory lecture. In particular, we are allowed to discuss the problems and solutions in this assignment, but have waited at least 30 minutes by doing other activities unrelated to class before attempting to complete or modify our answers as per the class policy.

We have documented our use of AI tools (if applicable) in a following table, as suggested in the NUS AI Tools policy[4]. This particular document did not use any AI Tools to proofcheck and was constructed and edited purely by manual work.

Signed, [A0196717N, A0205773A, A0219763W, A0268425X, A0236178X]

[4] https://libguides.nus.edu.sg/new2nus/acadintegrity, tab "AI Tools: Guidelines on Use in Academic Work"

Ethical Statement

Biases and Stereotypes Predictions generated by our models may include stereotypes that are untrue. Users should be aware of this risk.

For example, given the context of "A boy and a girl are sitting next to one another" and the question "Who has the bigger salary?", our model's answer was "Boy", possibly due to the prevalence of stereotypical depiction of gender roles in the training set.

Environmental Impact Most of our computation was done on Kaggle using GPU T4(x2). The total number of hours accumulated by our group can be estimated at around 80 hours. The carbon emission estimate can be computed using the Machine Learning Impact calculator[5].

Appendix

A Hyperparameter tuning for baseline model

1. Batch Size = 4 Although most literatures use the batch size of 16, we concluded that using a smaller batch size will lead to faster convergence due to more frequent updates, and thus better results for our usecase (small dataset and limited training time).

Batch Size	EM	F1
16	50.6%	0.637
4	**53.7%**	**0.683**

Table 10: Changing batch size for baseline model

2. Learning rate = 2e-5

   This learning rate is consistent with what other literatures report.

Learning Rate	EM	F1
2e-4	33.2%	0.452
2e-5	**53.7%**	**0.683**
2e-6	45.3%	0.580

Table 11: Changing learning rate for baseline model

   From the figure, we can see that the learning rate of 2e-5 yields the best result for our experiment setting.

[5] https://mlco2.github.io/impact/#compute, "Impact Calculator"