Project Report

Jian Mao
Department of Electrical Engineering
Columbia University
jm5134

Zihui Ouyang
Department of Statistics
Columbia University
zo2151

Jianfei Pan
Department of Mechanical Engineering
Columbia University
jp4201

***Abstract*—We are attempting to reproduce QANet, a deep network that aims to solve reading comprehension task based on SQuAD dataset.**

I. Overall Content

A. Review of Literature

The current end-to-end question and answering models are primarily based on Recurrent Neural Networks (RNNs). However, these methods are often slow and inefficient in training and inference due to its sequential nature that limits its ability for parallelization. Moreover, RNNs often can not model long term dependency effectively, although there have been some famous models for Natural Language Processing include the Gated Recurrent Unit [1] or Long Short Term Memory architectures [2].

There have also been attempts to use reinforcement learning to perform text classification models [3], but it is not certain whether this model is robust enough in tackling complicated tasks such as questions and answering. Moreover, attempts have been made to replace the RNNs completely with convolutional or attention layers, which have been proven to be faster and more effective than traditional RNNs. For example, the Bidirectional Attention Flow model uses attention mechanism and achieves state-of-art results on the Stanford Question Answering Dataset (SQuAD) [4]. The question and answering tasks in this paper often entails long texts that really cast challenges to the efficiency and accuracy of many current methods.

B. Description of the problem

The review of literature has shown that although there have been some successful and more efficient models by replacing RNN models with more efficient networks, there still lacks a both fast and accurate reading comprehension model. Moreover, there has not been a reading comprehension model that uses both convolutions and self-attention. Thus, the authors of this paper propose a new Q & A architecture that does not use RNNs but instead employs separable convolution and self-attention networks. The convolution models local interactions, whereas the self-attention models global interactions. The ANN model is fully feedforward and composed entirely of separable convolutions, attention, linear layers, layer normalization with residual connections in the encoder block. The dataset is Stanford Question Answering Dataset (SQuAD) version 1.1 with 107.7K query-answer pairs, with 87.5K for training, 10.1K for validation, and 10.1K for testing. Training and validation sets are available for public use, however test set is private. This dataset contains passages from Wikipedia and there are 5 questions for each of the passages. Overall, the author has demonstrated the model is both fast and accurate in the original paper, from which the team reproduced the model and achieved impressive results.

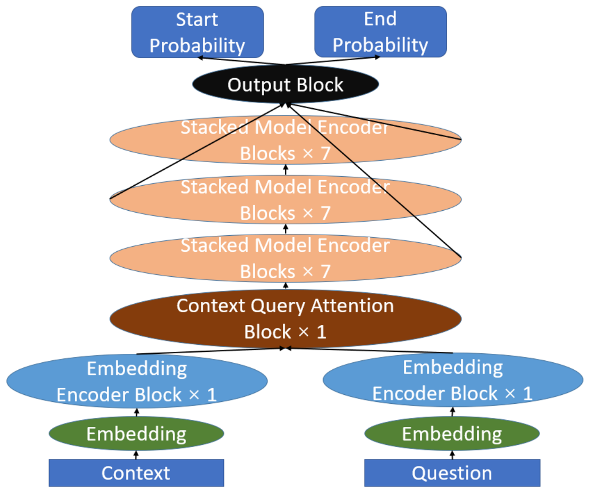
Start Probability
End Probability
Output Block
Stacked Model Encoder Blocks × 7
Stacked Model Encoder Blocks × 7
Stacked Model Encoder Blocks × 7
Context Query Attention Block × 1
Embedding Encoder Block × 1
Embedding Encoder Block × 1
Embedding
Embedding
Context
Question

Fig. 1. Overall architecture of model.

II. Description of Implementation

In this section, the team will dive into our detailed implementation that includes data preprocessing, detailed models, training process and parameter settings. The team will talk about similarity to the original paper and also changes from the original paper that we made to speed up our progress. The team will acknowledge that much of the design was borrowed from ewrfca's implementation on GitHub. (https://github.com/ewrfcas/QANet_keras). The EM score for a base model(with no data augmentation) was 73.6 and F1 score 82.7. In the paper it was also stated that there 3 hours were spent before 77.0 F1 score on validation set was achieved. With data augmentation(2 times the original size of training set), the EM increased to 74.5 and F1 score 83.2.

A. Model Summary

Our model overall is the same as the original QANet architecture with five main layers, input embedding layer,

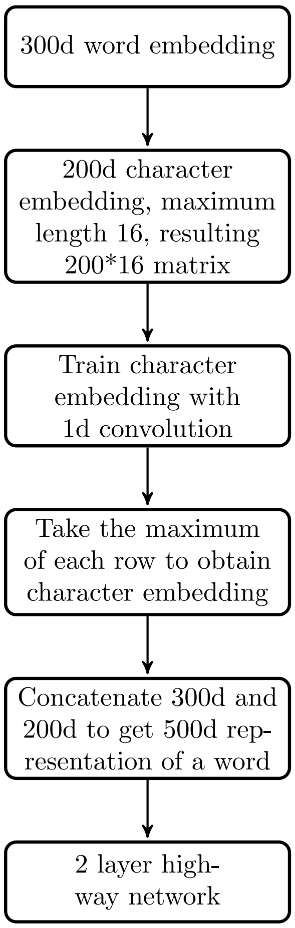
300d word embedding
200d character embedding, maximum length 16, resulting 200*16 matrix
Train character embedding with 1d convolution
Take the maximum of each row to obtain character embedding
Concatenate 300d and 200d to get 500d representation of a word
2 layer highway network

Fig. 2. Block diagram of embedding layer.

embedding encoder layer, context-query attention layer, model encoder layer, and output layer. Figure 1 has shown the details of our model. The context and questions are both passed into embedding layers first followed by embedding encoder blocks, the outputs of which will be fed into a context query attention block. The output will then be passed through three stacked model encoder blocks sequentially. The outputs of each block will be fed into the output block to yield the final output that consists of the input and output probabilities as well as the predicted location of the start and end locations.

B. Input embedding layer and data preprocessing

The team decided to discuss the data preprocessing together with the input embedding layer. This is because the team preprocessed the data before the embedding layer, which achieves the same effects as performing the same procedures in the original paper's input embedding layer. To improve speed we used the package tqdm to process data. The package is a smart progress bar for tracking the number of iterations, but it can also dramatically increase computational speed. We also chose not use nltk as suggested by the paper, but rather used WordPiece, a subword-based tokenization algorithm. Also, we did not attempt the data augmentation technique described in the paper since it is hard to implement and also it possibly

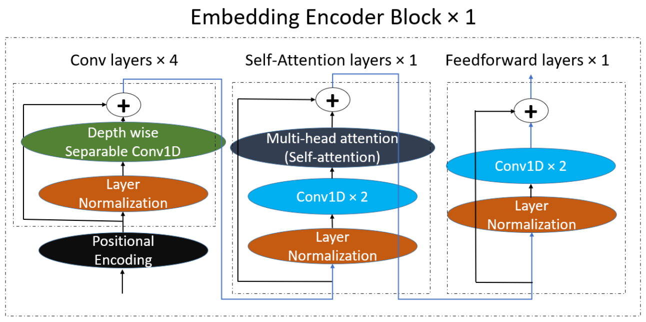

Fig. 3. Diagram of the encoder block

requires much time. Data augmentation in this case is to train two translation models, one from English to French, the other French to English, to obtain paraphrases.Our data preprocessing resembles the input embedding layer design in the original paper that we first concatenated the word embedding and character embedding for each word with a limit of 400 words in the contexts and 50 words in the questions [5]. We then padded the paragraphs to the limit size if the inputs are smaller than the limit. The dimension of the word embedding is 300 and of each character's embedding is 200. The team uses the pretrained GloVe embedding with 840B tokens and 2.2 million vocabulary as the word embedding. The embedding vectors were obtained directly from online sources (https://nlp.stanford.edu/projects/glove/). According to the paper, the word embedding is fixed during training, whereas the character embeddings is trainable. The team randomly initialized character embeddings from Normal distribution with variance 0.01 and concatenated them with word embeddings. Then, the embeddings will separately pass through a one dimensional convolution layer and a max pooling layer, on top of which we will pass them through a two-layer highway network. Highway networks were used possibly to increase depth of the network.

C. Context-query attention block

The outputs to this block are the encoded context, encoded questions and their masks. Their masks are calculated so that the padded portion of the sequence will not affect our attention calculations. The team's context to query attention block follows the structures proposed in the original paper by calculating the trilinear similarities between each pair of context and query words [5]. Each row of similarity matrix is then normalized by applying softmax functions, with which the context to query attention is calculated. We also compute the column normalized similarity matrix that is then used to compute the query to context attention matrix. The attention outputs will then be passed through a one dimensional convolution layer with 128 filters and a kernel size of 1. The outputs of this block are attention matrices A and B.

D. Model Encoder Block

Figure 5 shows the layout of the model encoder layer that is very similar to the input embedding layer. The only difference

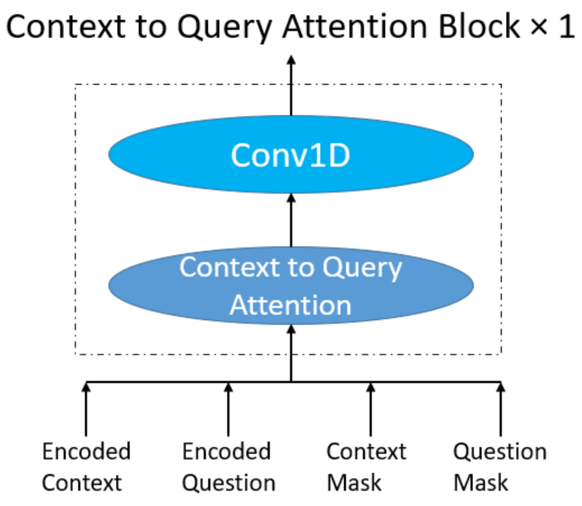
Context to Query Attention Block × 1
Conv1D
Context to Query Attention
Encoded Context
Encoded Question
Context Mask
Question Mask

Fig. 4. Diagram of the attention block.

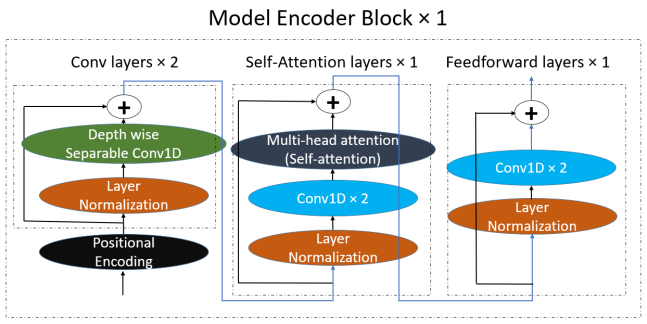

Fig. 5. Diagram of the model encoder block.

is that there are 2 convoluted layers instead of 4 from the input embedding layer. The inputs to this layer are the respective row of the attention matrices from the context to the query block. As shown by Figure 2, there are seven model encoder blocks in each stacked encoder blocks module, and the three stacked encoder blocks share weights.

E. Output Block

Figure 6 shows configuration of the output block. The inputs to this last module are outputs from the three stacked model encoder blocks shown in Figure 1.1. The outputs from the first and second block, and the outputs from meth first and third blocks are concatenated and processed separately. We adapted the same strategy of predicting the probability of each word in the context of being a start or end of the answer span [5]. The concatenated inputs are then passed through a feedforward network, which in our case was defined to a one dimensional convolution operation with softmax applied at each output to yield probabilities. The question and answer layer convert the outputs to find the most suitable prediction of the answer span, based on the constraints that the starting index is smaller than the end index with maximum likelihood. The raw probabilities will be padded to the text limit if results are somehow deprecated. In the end, the final outputs contain four terms that are the raw probabilities and predicted location of starting and ending index of the answer span, if the answer span can be found. The losses include two cross entropy losses for the first two outputs and two mean absolute error losses for the latter two outputs.

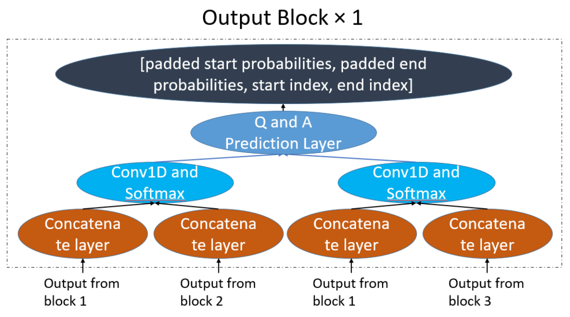

Fig. 6. Diagram of the output block.

F. Training and hyperparameter tuning

The batch size was set to 12, since large values may cause memory exhaustion problems and abort training.

In the second training, the learning rate warm-up scheme was not adopted, but maintained a constant learning rate of 0.001. This scheme was used in the third training. Exponential moving average was not applied in the three training sessions. Other hyperparameters remained consistent with the paper.

III. Results

A. Accuracy

Here, EM stands for exact match, measures the percentage of answers matches any of the ground truth exactly. F1 score measures the average overlap between prediction and ground truth. All the results are measured on validation set.

TABLE I
Training results compared to baseline

Model	**Summaries**	
	EM	***F1***
12 epochs without warming up	46.5	58.2
14 epochs with warming up	46.7	58.2
QANet Base	73.6	82.7

B. Training Times

The team trained the model three times. In the first time, the number of epochs was set to 6, and it took about 8.5 hours. In the second time, the number of epochs was set to 12, and it took about 17 hours. In the third time, the number of epochs is set to 14, and it took about 20 hours. The first training failed, and the last two succeeded.

C. Effort in optimizing results

The team tuned the parameters in order to obtain a model with better performance. Due to the lengthy running time, the team did not have enough time to try more hyperparameters.

The team hopes to continue tuning hyperparameters in the future. Moreover, the original paper has demonstrated impressive improvements by introducing augmented data. Thus, the team plans to try several data augmentation techniques such as manually replacing some texts with appropriate synonyms. We also plan to use google translate to transform our texts from English to other languages and then translate them back. Lastly, we can also pretrain or train another translation model using some state-of-the-art translation models to improve the diversity of our data. The author has pointed out the diversity of data is critical in improving the model performance.

D. Assumptions and results difference from the original paper

The original paper did not explicitly explain how the author designed the feedforward layers. Thus, we used stacked convolutional layers instead of dense layers to form our feedforward layers, because we find it frequently appears in similar models from online resources. The team customized the convolutional layer to have 128 filters and a kernel size of 5.

Also, we did not build and train another recurrent neural network to compare our model's time efficiency against for the same reasons.

IV. Discussion

A. Discussion of Results

Our results have shown that our model performed poorer than the original paper. This happens for a few reasons. Firstly, the paper was quite dense in information, and the author did not show details in all of his layers and procedures. Moreover, we found that online resources regarding the QANet are very limited and obscure. Thus, there was plenty of room for improvisation in designing our model. For example, we adopted convolutional layers as feedforward layers and have an arbitrary number of kernel sizes and filters for many convolutional layers. Therefore, the team thinks it is reasonable to have discrepancies between our results and those from the original paper.

Secondly, the team thinks the biggest upset from our results is due to insufficient training time and epoches. Because training 12 epochs took more than 12 hours on a local machine to complete, the team could not afford training our networks for days after we had used up our GCP services. We also could not afford tuning parameters extensively due to the time cost. Based on the results, it seems that the F1 scores of our model continue increasing after our designated epochs. Therefore, the team thinks our model has the potential to achieve a much better and closer F1 score to the paper's if we let it run for more epochs. Moreover, we did not implement the inverse exponential learning rate warm-up, which could influence our results as well.

Nevertheless, due to the limited time and resources, the team did not have enough time to reproduce every detail from the original paper. As discussed in the results section, the author trained multiple neural networks in this paper such as RNN based networks, translator networks and so on. The sheer volume of knowledge and strenuous training processes cannot be fully comprehended and reproduced by the team in such a short amount of time. The team aimed to perform more data augmentations techniques to enrich our dataset and retrain our model in the future. Based on comparison with the original paper, the data augmentation definitely helped improve the model performance significantly, which partially explained why our results are poorer than the author's.

B. Comparison with other papers

We are going to see two models that were developed around the same time as QaNet. The first model would be BERT(Bidirectional Encoder Representations from Transformers), which is a popular language generation model nowadays and also reads sentences in a bidirectional fashion rather than a unidirectional one. The original paper states that the result for a single base model is 80.8 for EM and 88.5 for F1 score [6]. The other one is the BiDAF(Bi-Directional Attention Flow) model [4], which is somewhat similar to QANet but it does not have blocks like QANet and it also uses RNN that QAnet does not. The single model produces 67.3 EM and 77.3 F1 score.

C. Discussion of problems faced

The team has faced numerous challenges during replicating the original paper and in our attempts to reproduce the results. Many challenges are frequently encountered in our data preprocessing, model construction, and model fitting.

1) Challenges in Data preprocessing: The team initially was kind of confused by the nested dictionary structures of the SQuAD dataset. The data preprocessing is very different from our prior experience in dealing with images. It took the team days to understand the structures of the SQuAD dataset and another few days to come up with strategies to preprocess the data. When we started preprocessing the dataset by converting the words into embedding and initializing character embeddings, we realized that data preprocessing takes a long time and the total dataset could easily exceed 80 GB with our chosen GloVe embedding vectors. It posed serious challenges to our data storage and training process, since we could not easily transfer data between computers. Eventually, we decided to run all data preprocessing and train our models on the one same computer. Moreover, we initially selected the smallest GloVe embedding dataset hoping to speed up our process, but we soon realized that the small dataset lacks key vocabularies and doesn't differentiate capital letters. We then switched to a larger dataset at the cost of a much slower preprocessing process. On of that, we initially used NLTK tokenizer to tokenize our sentences into words, but soon we discovered that NLTK has difficulty tokenizing certain characters and symbols such as the quotation marks. We then switched to spacy tokenizer that solved this problem.

2) Challenges in Model Construction: The team encountered many difficulties in building functional layers and models, since there are many layers such as highway layers, trilinear similarities and so on, which are not in the tensorflow keras repository. Thus, the team needs to build those layers from scratch. The team spent lots of time studying and testing ways to build customized layers and customized models from the tensorflow tutorial website and online resources. During this process, the biggest challenge was to build trainable weights and biases with correct shapes and initialization, and use tensorflow built-in functions to achieve required data manipulation such as matrix transformations that are necessary for our customized layers. Nevertheless, even with customized layers, there are still challenges to build the QANet model by subclassing. The team improvised on building certain blocks such as the input embedding block and output block. In order to call each block of stacked layers, the team has to build customized functional calls that call on stacks of layers and perform skip connections. Overall, constructing our model incurred lots of trials and errors and challenges, as the team learned to build and synthesize different layers, blocks, and other transformations with tensorflow tools.

3) Challenges in Training: In total, three training sessions were conducted. The first time was an attempted training, with the number of epochs setted to 6, and during the training, it was found that the loss did not decrease, while EM and F1 scores did not increase, but kept fluctuating. After checking the code, we noticed that an important step was missed in the initialization of the model class, i.e., the customized layers should be initialized before they are called, resulting in the parameters of each layer of the model not being optimized during training. For the second training, 12 epochs were run with a fixed initial learning rate of 0.001, and the learning rate warm-up scheme in the paper was not used. For the third training, 14 epochs were conducted and the warm-up scheme was adopted. The latter two training achieved correct data and results. Another issue in the training process is the hardware device, unlike the NVIDIA p100 GPU used in the paper, an NVIDIA RTX 3080 Laptop GPU was used. Due to hardware performance limitations, the batch size could only be set to a maximum of 12, and if set to a larger value, memory exhaustion may occur, making the training interrupted. Since the model is large and complex, each training epoch takes about 85 minutes, and we do not try more hyperparameters for training.

V. Conclusion

In this paper, we reproduced the QANet model [5], and aimed to reproduce their results. QANet is the first network that does not employ any RNN nature layers but only convolutional layers and attention layers. QANet is expected to train much faster while achieving state-of-art accuracies on SQuAD dataset. We customized layers and built the custom QANet model using subclassing with tensorflow tools and tensorflow functional API. Based on our current results, we are confident that the model works as expected. We tried improving the model's performance but the training process is overly slow due to the complicated model structure and large input data. Thus, the team believes our model can achieve comparable performance with the author's QANet, if we could train more epochs and tune various parameters. The team also believes data augmentation can significantly improve the model performance. In the future, we would like to try using translations and paraphrasing existing datasets to enrich and augment our dataset and fit our model with more diverse data. Most importantly, we would like to train our model with early stopping and for a few days straight to further analyze if there is any significant discrepancy between our QANet model and the one proposed in the original paper.

VI. Acknowledgement

We would like to thank GitHub user ewrfcas for his implementation that this group takes much inspiration from. We would also like to thank TA Feroz Ahmad for helping us.

VII. Contribution of Team Members

	jm5134	zo2151	jp4201
Last Name	Mao	Ouyang	Pan
Fraction of work	1/3	1/3	1/3
Work of 1	Training the model		
Work of 2		Find solutions to potential bugs	
Work of 3			Preprocess data Compiling Report

References

[1] J. Chung, et. al, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," https://arxiv.org/abs/1406.1078.

[2] S. Hochreiter, and J. Schmidhuber, "Long short-term memory," Neural Computation (1997) 9 (8): 1735–1780.

[3] A. W. Yu, H. Lee and Q. V. Le, "Learning to skim text," In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pp. 1880–1890, 2017.

[4] M. Seo, A. Kembhavi, A. Farhadi and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," https://arxiv.org/abs/1611.01603.

[5] A. W. Yu, et al., "QANet: combining local convolution with global self-attention for reading comprehension," https://arxiv.org/abs/1804.09541.

[6] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," https://arxiv.org/abs/1810.04805.