FAST AND ACCURATE READING COMPREHENSION WITHOUT RECURRENT NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Current end-to-end machine reading and question answering (Q&A) models are primarily based on recurrent neural networks (RNNs) with attention. Despite their success, these models are often slow for both training and inference due to the sequential nature of RNNs. We propose a novel Q&A model that does not require recurrent networks yet achieves equivalent or better performance than existing models. Our model is simple in that it consists exclusively of attention and convolutions. We also propose a novel data augmentation technique by paraphrasing. It not only enhances the training examples but also diversifies the phrasing of the sentences, which results in immediate accuracy improvements. This technique is of independent interest because it can be readily applied to other natural language processing tasks. On the SQuAD dataset, our model is 3x to 13x faster in training and 4x to 9x faster in inference. Our single model achieves 82.2 F1 score on the development set, which is on par with best documented result of 81.8.

1 INTRODUCTION

There is growing interest within natural language processing and machine learning communities in the tasks of machine comprehension and question answering. In the past few years, significant progress has been made in this area, with end-to-end models showing promising results on many challenging datasets. The most successful models generally employ two key components: (1) a recurrent model to process sequential inputs, and (2) an attention component to cope with long term interactions. For instance, the use of bi-attention model (Seo et al., 2016) for the SQuAD dataset (Rajpurkar et al., 2016). A weakness of these models is that they are often slow for both training and inference due to their recurrent nature, especially for long texts. The expensive training leads to high turnaround time for experimentation and limits researchers from rapid iteration, while the slow inference prevents the machine comprehension systems from scaling up and being deployed in real-time applications.

In this paper, aiming to make the machine comprehension both fast and accurate, we propose to remove the recurrent nature of these models. We instead exclusively use convolutions and self-attentions everywhere as the building blocks of encoders, while we learn the interactions between context and question by normal attentions(Bahdanau et al., 2015). Our model separately encodes the query and context, uses standard context-to-query attention to combine the two streams, then encodes the result before finally decoding to the probability of each position being the start or end of the answer span. Each encoder block is composed of convolutions and self attention. This architecture is illustrated in 1.

The underlying principle of our model is the following: The convolutions captures the local structure of the text, while the self-attention mechanisms learns the global interaction between each pair of words. The additional context-to-query attention is a standard module to construct the query-aware context vector for each position in the context paragraph, which is used in the subsequent modeling layers. The lack of recurrence speeds up our model significantly. In our experiments on the SQuAD dataset, our model is 3x to 13x faster in training and 4x to 10x faster in inference. In addition to the speed increase, the model is also competitive with previous models in terms of prediction quality. Besides, we propose a new data augmentation methodology to enhance the training data, which further improves the testing accuracy of our model. On the SQuAD dataset, our model achieves

82.2 F1 score on the development set [1], which is on par with the best published result of 81.8 by Hu et al. (2017). We also conduct ablation test to justify the usefulness of each component of our model. To the best of our knowledge, this is the first work on speeding up the machine comprehension tasks by removing recurrent components, while simultaneously boosting the prediction quality.

The contribution of this paper can be summarized as follows:

- We propose a simple, efficient, and effective reading comprehension model that exclusively built upon convolutions and attentions. It achieves up to 13x speedup in training and 9x in inference, compared to the RNN counterparts.
- We propose a novel data augmentation technique to enrich the training data by paraphrasing. It allows the model to achieve significantly higher performance that is on par with state-of-the-art.

2 THE MODEL

2.1 PROBLEM FORMULATION

The reading comprehension task considered in this paper, is defined as follows. Given a context paragraph with n words $C = \{c_1, c_2, ..., c_n\}$ and the query sentence with m words $Q = \{q_1, q_2, ..., q_m\}$, output a span $S = \{c_i, c_{i+1}, ..., c_{i+j}\}$ from the original paragraph C. In the following, we will use x to denote both the original word and its embedded vector, for any $x \in C, Q$.

2.2 MODEL OVERVIEW

The high level structure of our model is similar to the existing simple models such as FastQAExt (Weissenborn et al., 2017), which contains three major components: an embedding encoder, a context-to-question attention module, and a modeling encoder, as illustrated in Figure 1. These are the standard building blocks for most, if not all, existing reading comprehension models. However, the major differences between our approach and the existing ones are the following:

- For both the embedding and modeling encoders, we only use convolutional and self-attention mechanism, completely discarding RNNs, which are used by *ALL* existing reading comprehension models. As a result, our model is much faster, as it can process the input tokens in parallel.
- Our model is simpler than most in that it does not contain a complicated interactive attention layer between context and query, such as bi-attention (Seo et al., 2016), coattention (Xiong et al., 2016), gated attention (Wang et al., 2017; Yang et al., 2016), multi-hop reading (Hu et al., 2017; Shen et al., 2017b; Gong & Bowman, 2017).
- Our model does not rely on the addition of syntactic features such as POS tagging or name entity recognition, which have been used in Chen et al. (2017); Liu et al. (2017); Hu et al. (2017).

More specifically, our model consists of the following layers:

1. Input Embedding Layer. We adopt the standard techniques to obtain the embedding of each word w by concatenating its word embedding and character embedding. The word embedding is fixed during training and initialized from the $p = 200$ dimensional pre-trained GloVe (Pennington et al., 2014) word vectors, which are fixed during training. All the out-of-vocabulary words are mapped to an <UNK> token, whose embedding is trainable with random initialization. The character embedding is obtained as follows: Each character is represented as a trainable vector of dimension p, meaning each word can be viewed as the concatenation of the embedding vectors for each of its characters. We take maximum value of each row of this matrix to get a fixed-size vector representation of each word. Finally, the output of a given word x from this layer is the concatenation $[x_w; x_c] \in \mathbf{R}^{2p}$, where $p = 200$, x_w and x_c are the word embedding and the convolution output of character embedding of x respectively. For simplicity, we also use x to denote this embedded vector.

[1] At the time of submission, our model is being submitted for evaluation on the official test set

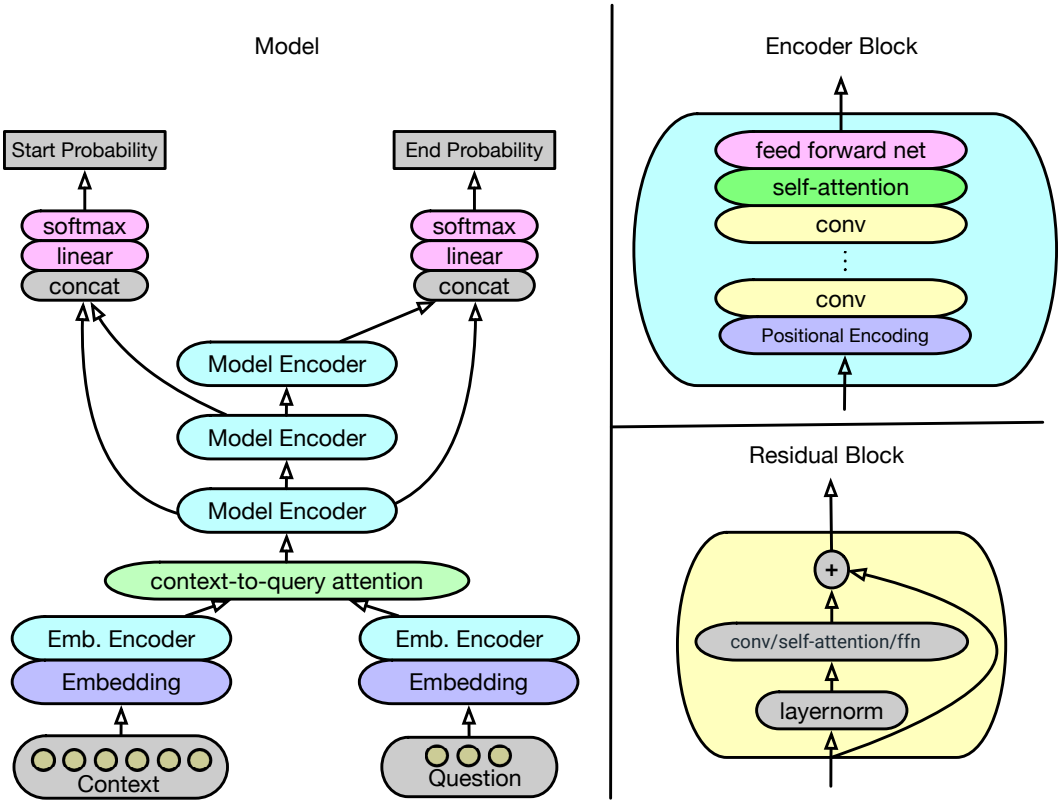

Figure 1: An illustration of our model architecture. We use the same encoder structure throughout the model, only varying the number of convolutional layers for the embedding encoders and model encoders. We additionally share weights of the context and question encoder, and of the three output encoders. A positional encoding is added to the input at the beginning of each encoder layer consisting of sin and cos functions at varying wavelengths, as defined in (Vaswani et al., 2017a). Each sub-layer after the positional encoding (one of convolution, self-attention, or feed-forward-net) inside the encoder structure is wrapped inside a residual block.

2. Embedding Encoder Layer. The encoder layer is a stack of the following basic building block: [convolution-layer $\times$ # + self-attention-layer + feed-forward-layer], as illustrated in the upper right of figure 1. We use depthwise separable convolutions (Chollet, 2016) (Kaiser et al., 2017) rather than traditional ones, as we observe that it is memory efficient, has better generalization, and allows us to use large kernel sizes. The kernel size is 7, the number of filters is $d = 128$ and the number of conv layers within a block is 4. For the self-attention-layer, we adopt the multi-head attention mechanism defined in (Vaswani et al., 2017a) which, for each position in the input, called the query, computes a weighted sum of all positions, or keys, in the input based on the similarity between the query and key as measured by the dot product. Each of these basic operations (conv/self-attention/ffn) is placed inside a *residual block*, shown lower-right in figure 1. For an input x and a given operation f, the output is $f(layernorm(x)) + x$, meaning there is a full identity path from the input to output of each block, where layernorm indicates layer-normalization proposed in (Ba et al., 2016). The total number of encoder blocks is 1. Note that the output of this layer is a vector of dimension $d = 128$, for each individual input word.

3. Context-to-Query Attention Layer. This module is standard in almost every previous reading comprehension models such as Weissenborn et al. (2017) and Chen et al. (2017). We use C and Q to denote the encoded context and query. The context-to-query attention is constructed as follows: We first computer the similarities between each pair of context and query words, rendering a similarity matrix $S \in \mathbf{R}^{n \times m}$. We then normalize each row of S by applying the softmax function, getting a matrix $\bar{S}$. Then the context-to-query attention is computed as $A = \bar{S} \cdot Q' \in \mathbf{R}^{n \times d}$. The similarity

function used here is the trilinear function (Seo et al., 2016):

$$f(q, c) = W_0[q, c, q \odot c],$$

where $\odot$ is the element-wise multiplication and W_0 is a trainable variable.

Most high performing existing models additionally use some form of query to context attention. However, we found that, when self attention layers are used, the addition of query-to-context attention provide little benefit over simply applying context-to-query attention.

4. Model Encoder Layer. Similar to Seo et al. (2016), the input of this layer at each position is $[c, a, c \odot a]$, where a is a row of attention matrix A. This layer is similar to the Embedding Encoder Layer except that convolution layer number is 2 within a block and the total number of blocks are 7. We share weights between each of the 3 repetitions of the model encoder.

5. Output layer. This layer is task-specific. Each example in SQuAD is labeled with a span in the context containing the answer. We adopt the strategy of Seo et al. (2016) to predict the probability of each position in the context being the start or end of an answer span. More specifically, the probabilities of the starting and ending position are modeled as

$$p^1 = softmax(W_1[M_0; M_1]), \;\; p^2 = softmax(W_2[M_0; M_2]),$$

where W_1 and W_2 are two trainable variables and M_0, M_1, M_2 are respectively the outputs of the three model encoders, from bottom to top. The score of a span is then the product of its start position and end position probabilities. Finally, the objective function is defined as the negative sum of the log probabilities of the true start and end indices by the predicted distributions, which is averaged over all the training examples:

$$L(\theta) = -\frac{1}{N} \sum_{i}^{N} \left[\log(p^1_{y^1_i}) + \log(p^2_{y^2_i}) \right],$$

where y^1_i and y^2_i are respectively the groundtruth starting and ending position of example i, and θ contains all the trainable variables. The proposed model can be customized to other comprehension tasks, e.g. selecting from the candidate answers, by changing the output layers accordingly.

Inference. At inference time, the predicted span (s, e) is chosen such that $p^1_s p^2_e$ is maximized and $s \leq e$. Standard dynamic programming can obtain the result with linear time.

3 Data Augmentation by Paraphrasing

Orthogonal to the model design, we also propose a novel data augmentation methodology to enhance the training data size, which results in further performance gains in the testing accuracy. Our approach leverages the off-the-shelf neural machine translation (NMT) models. We place the original context paragraph into a NMT system to obtain the translated context in a new language, e.g. French, which is translated back to English to get the paraphrase. The process is illustrated in Figure 2.

Two NMT models are used for paraphrasing, one for translating from English to French and one for back translating. The high level architecture is similar to the PARANET with k-best translations proposed by Lapata et al. (2017), but without any modification to the decoder. We use the codebase from Tensorflow NMT tutorial (Luong et al., 2017) with the 4 layers GNMT (Wu et al., 2016) configuration[2] prepared by the tutorial for both models. Models are trained with WMT14 English-French parallel corpus[3], which is tokenized and segmented into subword units. For the SQuAD dataset, paragraphs are split into sentences, which are further tokenized and segmented into subword units with the same configuration as NMT models. We keep the top 5 outputs for each translation from beam search; finally, 25 paraphrases are generated for each sentence in the original SQuAD dataset.

[2] https://github.com/tensorflow/nmt/blob/master/nmt/standard_hparams/wmt16_gnmt_4_layer.json

[3] http://www.statmt.org/wmt14/

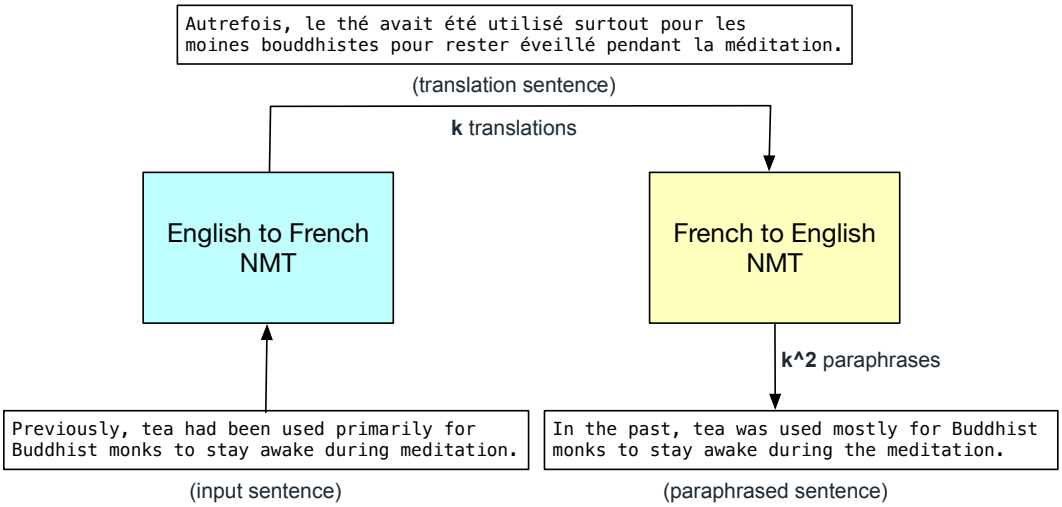

Figure 2: An illustration of the data augmentation process. **k** is the beam width, which is the number of translations generated by the NMT system.

We sample a paraphrase for each sentence in the original paragraph to form a new paragraph. This process is applied to every paragraph and question pair. An obvious issue with this approach is the answer in source sentences being paraphrased to a similar but different phrase. If the original sentence contains an answer, we try to identify the paraphrased answer in the sampled paraphrase, and use it as the new answer. We use a simple heuristic to find the answer and eliminate paraphrases without clear matches. We calculate character level 2-gram match scores to find possible start and end positions of an answer, and then calculate the match score for the original answer with all selected phrases in the paraphrase. The phrase with the highest score is selected as the new answer. Table 1 shows an example of the new answer found by this process. We also define a minimum threshold for elimination. If there is no answer with score higher than the threshold, we remove the paraphrase from our sampling process. If all of 25 paraphrases for a sentence are eliminated, no sampling will be performed for the sentence.

	Sentence that contains an answer	Answer
Original	All of the departments in the College of Science offer PhD programs, except for the Department of Pre-Professional Studies.	Department of Pre-Professional Studies
Paraphrase	All departments in the College of Science offer PHD programs with the exception of the Department of Preparatory Studies.	Department of Preparatory Studies

Table 1: Comparison between answers in original sentence and paraphrased sentence.

The quality and diversity of paraphrases are essential to the data augmentation method. It is still possible to improve the quality and diversity of this method. The quality can be improved by using better translation models. For example, we find paraphrases significantly longer than our models' maximum training sequence length tend to be cut off in the middle. The diversity can be improved by both sampling during the beam search decoding and paraphrasing questions and answers in the dataset as well. In addition, we can combine this method with other data augmentation methods, such as, the type swap method (Raiman & Miller, 2017), to acquire more diversity in paraphrases.

In our experiments, we observe that the proposed data augmentation can bring non-trivial improvement in terms of accuracy. We believe this technique is also applicable to other supervised natural language processing tasks, especially when the training data is insufficient.

4 EXPERIMENTS

In this section, we conduct experimental studies to test the effectiveness and efficiency of our model.

4.1 DATASET

The test bed is the Stanford Question Answering Dataset (SQuAD)[4] (Rajpurkar et al., 2016) for machine comprehension. The dataset is constructed using 536 articles randomly sampled from Wikipedia. Each training example consists of a context paragraph of those articles and an associated query, and the answer must be a span from the paragraph. SQuAD contains 107.7K query-answer pairs, with 87.5K for training, 10.1K for validation, and another 10.1K for testing. The typical length of the paragraphs is around 250 while the question is of 10 tokens. However, there are exceptionally long cases. Only the training and validation data are publicly available, while the testing data is hidden that one has to submit the code to a colab and cope with the authors of (Rajpurkar et al., 2016) to retrieve the final testing score. For the time being, we only show the performance on the validation set, as our code is being submitted for official evaluation. According to the previous works, such as (Seo et al., 2016; Xiong et al., 2016; Wang et al., 2017; Chen et al., 2017), the testing score is very close to and always a bit higher than the validation score, so we believe the comparison is convincing.

4.2 BASIC SETUP

Data Preprocessing We use the NLTK tokenizer[5] to preprocess the data. During training, the context length is set to 400. Any paragraph longer than that would be discarded and the short ones are pad with special symbol `<PAD>`. The maximum answer length is set to 30. We use the pre-trained 200-D word vectors GLoVe (Pennington et al., 2014), and all the out-of-vocabulary words are replace with `<UNK>`, whose embedding is updated during training. Each character embedding is randomly initialized as a 200-D vector, which is updated in training as well. We choose two augmented datasets obtained from Section 3, which contain 140K and 240K examples and are denoted as "data aug x2" and "data aug x3" respectively, including the original data.

Model Parameters The hidden size and the convolution filter number are all 128, the batch size is 32, training steps are 120K for original data, 200K for data aug x2, and 250K for data aug x3. The numbers of convolution layers in the embedding and modeling encoder are 4 and 2, kernel sizes are 5 and 7, and the block numbers for the encoders are 1 and 7, respectively. The word and character dropout are 0.1 and 0.05 respectively, and we apply dropout between every two layers of 0.1.

Optimization Details We adopt the ADAM (Kingma & Ba, 2014) optimizer with $\beta_1 = 0.8, \beta_2 = 0.999, \epsilon = 10^{-7}$. We use a learning rate warm-up scheme with an inverse exponential increase from 0.0 to 0.001 in the first 1000 steps, and then maintain a constant learning rate for the remainder of training. Exponential moving average is applied on all trainable variables with a decay rate 0.9999.

Platform All the codes are written with python using the toolkit Tensorflow[6] (Abadi et al., 2016) and run on a NVIDIA p100 GPU.

4.3 THE RESULTS AND ANALYSIS

Accuracy The F1 and Exact Match (EM) are two evaluation metrics of accuracy for the model performance. F1 measures the portion of overlap tokens between the predicted answer and groundtruth, while exact match score is 1 if the prediction is exactly the same as groundtruth or 0 otherwise. We show the result in Table 2. To make a fair and thorough comparison, we both report the published results in their latest papers/preprints and the updated but not documented results on the leaderboard. We deem the latter as the unpublished results. The accuracy (EM/F1) performance of our model is on par with the state-of-the-art models. In particular, our model outperforms all the documented results in the literature, in terms of the F1 score (see second column of Table 2).

[4] https://rajpurkar.github.io/SQuAD-explorer/

[5] http://www.nltk.org/

[6] https://www.tensorflow.org/

	Published[7]	LeaderBoard[8]
Single Model	EM / F1	EM / F1
LR Baseline (Rajpurkar et al., 2016)	40.4 / 51.0	40.4 / 51.0
Dynamic Chunk Reader (Yu et al., 2016)	62.5 / 71.0	62.5 / 71.0
Match-LSTM with Ans-Ptr (Wang & Jiang, 2016)	64.7 / 73.7	64.7 / 73.7
Multi-Perspective Matching (Wang et al., 2016)	65.5 / 75.1	70.4 / 78.8
Dynamic Coattention Networks (Xiong et al., 2016)	66.2 / 75.9	66.2 / 75.9
FastQA (Weissenborn et al., 2017)	68.4 / 77.1	68.4 / 77.1
BiDAF (Seo et al., 2016)	68.0 / 77.3	68.0 / 77.3
SEDT (Liu et al., 2017)	68.1 / 77.5	68.5 / 78.0
RaSoR (Lee et al., 2016)	70.8 / 78.7	69.6 / 77.7
FastQAExt (Weissenborn et al., 2017)	70.8 / 78.9	70.8 / 78.9
ReasoNet (Shen et al., 2017b)	69.1 / 78.9	70.6 / 79.4
Document Reader (Chen et al., 2017)	70.0 / 79.0	70.7 / 79.4
Ruminating Reader (Gong & Bowman, 2017)	70.6 / 79.5	70.6 / 79.5
jNet (Zhang et al., 2017)	70.6 / 79.8	70.6 / 79.8
Conductor-net	N/A	72.6 / 81.4
Interactive AoA Reader (Cui et al., 2017)	N/A	73.6 / 81.9
Reg-RaSoR	N/A	75.8 / 83.3
DCN+	N/A	74.9 / 82.8
AIR-FusionNet	N/A	76.0 / 83.9
R-Net (Wang et al., 2017)	72.3 / 80.7	76.5 /84.3
BiDAF + Self Attention + ELMo	N/A	**77.9/ 85.3**
Reinforced Mnemonic Reader (Hu et al., 2017)	73.2 / 81.8	73.2 / 81.8
Our Model + data aug x3	73.0 / **82.2**	N/A

Table 2: The performances of different models on SQuAD dataset.

Speedup To measure the speedup of our model against the RNN models, we also test the corresponding model architecture with each encoder block replaced with a stack of bidirectional LSTMs as is used in most existing models. Specifically, each 7-layer model encoder block is replaced with a 1, 2, or 3 layer Bidirectional LSTMs respectively, as such layer numbers fall into the usual range of the reading comprehension models (Seo et al., 2016; Chen et al., 2017). All of these LSTMs have hidden size 128. The results of the speedup comparison are shown in Table 3. We can readily see that our model is significantly faster than all the RNN based models and the speedups range from 3 to 13 times in training and 4 to 9 times in inference. Note that the RNN models tested here are much simpler than those proposed in previous works, since the we don't use the complicated bi-attention (Seo et al., 2016) or co-attention (Xiong et al., 2016) computation nor the multi-hop reading scheme (Shen et al., 2017b; Gong & Bowman, 2017). This implies our real-world speedup over these competitors to be even more significant.

	Ours	RNN-1-128	Speedup	RNN-2-128	Speedup	RNN-3-128	Speedup
Training	**3.2**	1.1	**2.9x**	0.34	**9.4x**	0.24	**13.3x**
Inference	**8.1**	2.2	**3.7x**	1.3	**6.2x**	0.92	**8.8x**

Table 3: Speed comparison between the proposed model and RNN-based models, all with batch size 32. RNN-x-y indicates an RNN with x layers each containing y hidden units. The RNNs used here are bidirectional LSTM. The processing speed is measured by steps/second, so higher is faster.

Ablation Study We conduct ablation studies on components of the proposed model, and investigate the effect of augmented data. The validation scores on the development set are shown in Table 4. The first take-home message is that the convolutions in the encoders are crucial: both F1 and EM drop drastically by 3 percent if it is removed. Self-attention in the encoders is also a nec-

[7]The scores are collected from the latest version of the documented related work on Oct 27, 2017.

[8]The scores are collected from the leaderboard on Oct 27, 2017.

Single Model	EM / F1
Full Model	72.5 / 81.4
- convolution in encoders	69.4 / 78.5
- self-attention in encoders	70.9 / 80.0
replaced with normal convolution	71.8 / 80.7
+ data aug $\times 2$	72.8 / 82.0
+ data aug $\times 3$	73.0 / 82.2

Table 4: Ablation study on different components and data augmentation. The result is based on model trained with original data unless stated otherwise.

essary component that contributes 1.4/1.6 gain of F1/EM to the ultimate performance. We interpret these phenomena as follows: the convolutions capture the local structure of the context while the self-attention is able to model the global interactions between text. Hence they are complimentary to but cannot replace each other. The use of separable convolutions in lieu of tradition convolutions also has a prominent contribution to the performance, which can be seen by the slightly worse accuracy caused by replacing with normal convolution. As the last two rows of the table show, data augmentation proves to be helpful in further boosting performance. $\times$ 2 data augmentation yields an increase in the F1 by 0.6 percent. The performance gain lessens with more data, as generating an additional 100K examples yields only another 0.2 improvement in F1. We observe that injecting more data beyond this amount does not benefit the model.

5 Related Work

Reading comprehension and question answering has become a very important topic in the NLP domain. Its popularity can be attributed to an increase in publicly available annotated datasets, such as SQuAD (Rajpurkar et al., 2016), CNN/Daily News (Hermann et al., 2015), WikiReading (Hewlett et al., 2016), Children Book Test (Hill et al., 2015), etc. A plethora of effective end-to-end neural network models exist to approach these challenges, including BiDaf (Seo et al., 2016), r-net (Wang et al., 2017), DCN (Xiong et al., 2016), ReasoNet (Shen et al., 2017b), Document Reader (Chen et al., 2017), Interactive AoA Reader (Cui et al., 2017) and Reinforced Mnemonic Reader (Hu et al., 2017).

Recurrent Neural Networks (RNNs) have been used extensively in the natural language processing area in the past few years. The sequential nature of the text coincides with the design philosophy of RNNs, and hence RNN is generally the default choice for modeling text. In fact, all the reading comprehension models mentioned above are based on RNNs. While effective, the sequential nature of RNN prevent parallel computation, as tokens must be fed into the RNN in order. Another drawback of RNNs is difficulty modeling long dependencies, although this is somewhat alleviated by the use of Gated Recurrent Units (Chung et al., 2014) or Long Short Term Memory cells. Reading comprehension considered in this paper always needs to deal with long text, as the context paragraphs may be hundreds of words long. Recently, attempts have been made to replace the recurrent networks by fully convolution or attention based architectures (Kim, 2014; Gehring et al., 2017; Vaswani et al., 2017b; Shen et al., 2017a). Those models have been shown to be not only faster than the RNN based ones, but also effective in different tasks, such as text classification, machine translation or sentiment analysis.

To our knowledge, our paper is the first work on both *fast* and *accurate* reading comprehension model, by discarding the recurrent networks in favor of feed forward architectures. Note that Raiman & Miller (2017) recently proposed to accelerate reading comprehension by avoiding bi-directional attention and making computation conditional on the search beams. Nevertheless, their model is still based on the RNNs and the accuracy is not competitive, with an EM 68.4 and F1 76.2. Weissenborn et al. (2017) also tried to build a fast Q&A model by deleting the context-to-query attention module. However, it again relied on RNN and is thus intrinsically slower than ours. The elimination of attention further has sacrificed the performance (with EM 68.4 and F1 77.1).

Data augmentation has also been explored in different scenarios of natural language processing. For example, Zhang et al. (2015) proposed to enhance the dataset by replacing the words with their

synonyms, which was shown to be effective in text classification. Recently, Raiman & Miller (2017) suggested using type swap to augment the SQuAD dataset, which essentially replaces the words in the original paragraph with others of exactly the same type. While it was shown to improve the accuracy, the augmented data has the same syntactic structure as the originals, so they are not sufficiently diverse. On the other hand, Zhou et al. (2017) aimed at improving the diversity of the SQuAD data by generating more questions. However, as reported by Wang et al. (2017), it did not help improve the performance. The data augmentation technique proposed in this paper is based on paraphrasing the sentences by translating the original text back and forth. The major benefit is that it can bring more syntactical diversity to the enhanced data. It also help our model to achieve state-of-the-art performance.

6 Conclusion and Discussion

In this paper, we propose a fast and accurate end-to-end model for machine reading comprehension. Our core innovation is to completely remove the recurrent networks in the base model. The resulting model is fully feedforward, composed entirely of separable convolutions, attention, linear layers, and layer normalization, which is suitable for parallel computation. We find that this results in a model that matches the best published results while up to 9 times faster than a comparable recurrent model during inference. Additionally, we find that we are able to achieve significant gains by utilizing data augmentation consisting of translating context and passage pairs to and from another language as a way of paraphrasing the underlying ideas. The mechanisms of our models is orthogonal to many existing variants. We believe that, when combined with more sophisticated techniques, it can obtain even higher performance. The data augmentation technique is general and of independent interest. We are convinced that many the NLP tasks can leverage it to obtain improvement in performance.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016. URL `http://arxiv.org/abs/1603.04467`.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL `http://arxiv.org/abs/1607.06450`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. 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. 1870–1879, 2017.

François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. URL `http://arxiv.org/abs/1610.02357`.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. 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. 593–602, 2017.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, 2017.

Yichen Gong and Samuel R. Bowman. Ruminating reader: Reasoning with gated multi-hop attention. *CoRR*, abs/1704.07415, 2017. URL http://arxiv.org/abs/1704.07415.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1693–1701, 2015.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. Wikireading: A novel large-scale language understanding task over wikipedia. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *CoRR*, abs/1511.02301, 2015.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. Reinforced mnemonic reader for machine comprehension. *CoRR*, abs/1705.02798, 2017. URL http://arxiv.org/abs/1705.02798.

Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*, 2017.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1746–1751, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Mirella Lapata, Rico Sennrich, and Jonathan Mallinson. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pp. 881–893, 2017. URL http://aclanthology.info/papers/E17-1083/paraphrasing-revisited-with-neural-machine-translation.

Kenton Lee, Tom Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. Learning recurrent span representations for extractive question answering. *CoRR*, abs/1611.01436, 2016.

Rui Liu, Junjie Hu, Wei Wei, Zi Yang, and Eric Nyberg. Structural embedding of syntactic trees for machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 826–835, 2017.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*, 2017.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word//w representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Jonathan Raiman and John Miller. Globally normalized reader. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 1070–1080, 2017.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 2383–2392, 2016.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016. URL http://arxiv.org/abs/1611.01603.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. *CoRR*, abs/1709.04696, 2017a. URL http://arxiv.org/abs/1709.04696.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pp. 1047–1055, 2017b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017a. URL http://arxiv.org/abs/1706.03762.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017b.

Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905, 2016. URL http://arxiv.org/abs/1608.07905.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. 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. 189–198, 2017.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. Multi-perspective context matching for machine comprehension. *CoRR*, abs/1612.04211, 2016. URL http://arxiv.org/abs/1612.04211.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making neural QA as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*, pp. 271–280, 2017.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016. URL http://arxiv.org/abs/1611.01604.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. Words or characters? fine-grained gating for reading comprehension. *CoRR*, abs/1611.01724, 2016. URL http://arxiv.org/abs/1611.01724.

Yang Yu, Wei Zhang, Kazi Saidul Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end reading comprehension with dynamic answer chunk ranking. *CoRR*, abs/1610.09996, 2016. URL http://arxiv.org/abs/1610.09996.

Junbei Zhang, Xiao-Dan Zhu, Qian Chen, Li-Rong Dai, Si Wei, and Hui Jiang. Exploring question understanding and adaptation in neural-network-based question answering. *CoRR*, abs/1703.04617, 2017.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 649–657, 2015.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural question generation from text: A preliminary study. *CoRR*, abs/1704.01792, 2017. URL http://arxiv.org/abs/1704.01792.