FAST READING COMPREHENSION WITH CONVNETS

Felix Wu*
Department of Computer Science
Cornell University
Ithaca, NY, USA
fw245@cornell.edu

Ni Lao, John Blitzer
Google Inc.
Mountain View, CA, USA
{nlao,blitzer}@google.com

Guandao Yang, Kilian Q. Weinberger
Cornell University
Ithaca, NY, USA
{gy46,kqw4}@cornell.edu

ABSTRACT

State-of-the-art deep reading comprehension models are dominated by recurrent neural nets. Their sequential nature is a natural fit for language, but it also precludes parallelization within an instances and often becomes the bottleneck for deploying such models to latency critical scenarios. This is particularly problematic for longer texts. Here we present a convolutional architecture as an alternative to these recurrent architectures. Using simple dilated convolutional units in place of recurrent ones, we achieve results comparable to the state of the art on two question answering tasks, while at the same time achieving up to two orders of magnitude speedups for question answering.

1 INTRODUCTION

Recurrent neural networks such as LSTMs (Hochreiter & Schmidhuber, 1997) and GRUs (Cho et al., 2014) are been very successful at simplifying certain natural language processing (NLP) systems, such as language modeling and machine translation. The dominant of *deep text understanding* models (Rajpurkar et al., 2016; Joshi et al., 2017; Seo et al., 2017) typically relies on recurrent networks to produce initial representations for the question and the document, and then apply attention mechanisms (Bahdanau et al., 2014) to allow information passes between the two representations. The recurrent units are powerful structures capable of modeling complex long range interactions. However, their sequential nature precludes parallelization within training examples, and often become the bottleneck for deploying models to latency critical NLP applications. High latency is especially critical for interactive question answering (for example as part of search engines or mobile assistants), as it requires the user to wait patiently for the answer.

Recent development of "attention only" deep text models Parikh et al. (2016); Vaswani et al. (2017) in various tasks allows modeling of long range dependencies without regard to their distance. By parallelization within one instance, these models can have much better inference time than those which depend on recurrent units. However, their token-pair based attention requires $O(n^2)$ memory consumption within the GPUs, where n denotes the length of the document. This quadratic growth prevents their use with most real-world documents, such as e.g. Wikipedia pages (e.g., Figure 10 compares the memory usage of different types of models).

*A majority of the work was done while the author was interning at Google.

Q: Plato and Xenophon were both pupils of which Greek philosopher?
Socrates ... philosophy. He is an enigmatic figure known chiefly through the accounts of classical writers, especially the writings of his students Plato and Xenophon and the plays of his contemporary Aristophanes. ...

Q: What is the next in the series: Carboniferous, Permian, Triassic, Jurassic?
... The Jurassic North Atlantic Ocean was relatively narrow , while the South Atlantic did not open until the following **Cretaceous** period , when ...

Q: Who was the choreographer of the dance troupe Hot Gossip?
Arlene Phillips, ... Lee Mack. Hot Gossip In Britain, Phillips first became a household name as the director and choreographer of Hot Gossip, a British dance troupe which she formed in 1974 ...

Figure 1: Samples from TriviaQA (Joshi et al., 2017)

In this work we propose, Gated Linear Dilated Residual Network (GLDR), a different architecture to avoid recurrent units in text precessing. More specifically, we use a combination of residual networks (He et al., 2016), dilated convolutions (Yu & Koltun, 2016) and gated linear units (Dauphin et al., 2017).

1.1 Reading Comprehension Tasks

Reading comprehension tasks focus on one's ability to read a piece of text and subsequently answer questions about it (see TriviaQA examples in Figure 1). We follow the typical reading compression setting and assume that the correct answer can be given as a snippet of the original text. This reduces the problem to a search problem, where the question functions as a query.

Sometimes these tasks simply involve answer type and query term matching, but they sometimes may contain discourse phenomena like coreference (e.g. *What philosopher taught Plato and Aristophanes?* and *Who was the choreographer of the dance troupe Hot Gossip?*) or even real world knowledge (e.g., answer *What is the next in the series: Carboniferous, Permian, Triassic, Jurassic?* potentially involves understanding the semantics of *next* and *following*). Figure 2 shows an adversarial example of a question that is answered incorrectly by matching the first occurrence of the query word "composed" in the answer text. This study will use two popular reading comprehension tasks – Trivia QA (Joshi et al., 2017), and SQUAD (Rajpurkar et al., 2016) – as its test bed. Both tasks have openly available training and validation data sets and are associated with competitions over a hidden test set on a public leaderboard.

Because of the sequential nature of documents and text, and complex long-distance relationships between words, recurrent neural networks (especially LSTMs Hochreiter & Schmidhuber (1997) and GRUs Cho et al. (2014)) are a natural class for modeling reading comprehension. Indeed, on both reading comprehension tasks we study here, every published result on the leader-board [1] [2] uses

[1] `https://competitions.codalab.org/competitions/17208`
[2] `https://rajpurkar.github.io/SQuAD-explorer`

Q: Who composed the works The Fountains of Rome and The Pines of Rome in 1916 and 1924 respectively?
Adversary example . Wolfgang Amadeus Mozart composed The Magic Flute , and Requiem . **Ottorino Respighi** (; 9 July 1879 - 18 April 1936) was an Italian violinist , composer and musicologist , best known for his three orchestral tone poems Fountains of Rome (1916) , Pines of Rome (1924) , and Roman Festivals (1928).

Figure 2: DrQA gives a wrong answer "Wolfgang Amadeus Mozart" rather than "Ottorino Respighi" by simply matching the verb "composed".

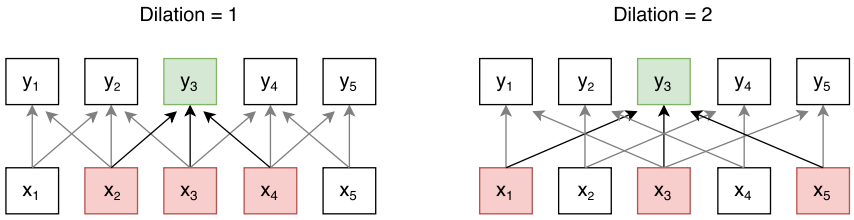

Figure 3: An illustration of dilated convolution. With a dilation of 1 (*left*), dilated convolution reverts to standard convolution. With a dilation of 2 (*right*) every other word is skipped, allowing the output y_3 to relate words x_1 and x_5 despite their large distance and a relatively small convolutional kernel.

some kind of recurrent mechanism. Below, we will discuss two specific models in detail, but we begin by motivating a one-dimensional convolution architecture for the reading comprehension task.

1.2 Text Understanding with Dilated Convolutions

Bidirectional recurrent units can in theory model arbitrarily long dependencies in text, but in practice we may be able to capture these dependencies through other mechanisms. We propose to substitute complicated and costly sequential models through simple feed-forward network architectures. There are two important criteria of language that LSTMs model, that we also want to capture. First, we may need to model relationships between individual words, even when they are separated from each through many words (e.g. Figure 1). Second, we want to model the compositional nature of natural language semantics, where the meaning of large phrases are composed of the meaning of their sub-phrases.

These constraints lead us to choose dilated convolutional networks (Yu & Koltun, 2016) with gated linear units (Dauphin et al., 2017). By increasing the receptive field in our convolutional units, dilation can help to model arbitrarily long-distance dependencies. Unfortunately, the receptive region is pre-determined, which prevents us from examining long range dependencies in detail. For instance, in the co-reference examples from Figure 1, we would need to directly convolve representations for "his" and "Socrates", but an increasing dilation will miss this. In practice, "Socrates" is combined with its context to give a fixed size representation for a long context. We alleviate some of this effect by using Gated Linear Units (Dauphin et al., 2017) in our convolutions. These units allow us to selectively retain (and compute gradients for) important features of low-level words and phrases, even at convolutions with larger dilations.

Dilated Convolution. Given a 1-D convolutional kernel $\mathbf{k} = [k_{-l}, k_{-l+1}, ..., k_l]$ of size $2l + 1$ and the input sequence $\mathbf{x} = [x_1, x_2, ..., x_n]$ of length n, a d dilated convolution of $\mathbf{x}$ with respect the kernel $\mathbf{k}$ can be described as

$$(\mathbf{k} * \mathbf{x})_t = \sum_{i=-l}^{l} k_i \cdot x_{t-d \cdot i}$$

where $t \in \{1, 2, \cdots, n\}$. Here we assume zero-padding, so tokens outside the sequence will be treated as zeros. Unlike normal convolutions (i.e. $d = 1$) that convolve each contiguous subsequence of the input sequence with the kernel, dilated convolution uses every d^{th} element in the sequence, but shifting the input by one at a time. Figure 3 shows an example of dilated convolution. Here, the green output is a weighted combination of the red input words.

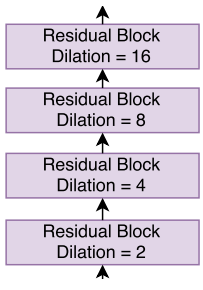

Figure 4: The receptive field of repeated dilated convolution grows exponentially with network depth.

Why Dilated convolution? Repeated dilated convolution (Yu & Koltun, 2016) increases the receptive region of ConvNet outputs exponentially with respect to the network depth, which results in drastically shortened computation paths. See Figure 4 for an illustration of an architecture with four dilated convolutional layers with exponentially increasing dilations. Table 1 shows a brief comparison between bidirectional recurrent units, self-attention, and dilated convolution. Self-attention suffers from the fact that the overall computation is quadratic with respect to the sequence length n. This may be tolerable in settings like machine translation, where a typical document consists

Layer type	Computations per layer	Minimum depth D to cover length n	Longest computation path	Overall computations
Recurrent Units	$O(w^2 n)$	$O(1)$	$O(nD)$	$O(w^2 n)$
Self-Attention	$O(wn^2)$	$O(1)$	$O(D)$	$O(wn^2)$
Dilated Convolution	$O(kw^2 n)$	$O(\log(n))$	$O(D)$	$O(kw^2 nD)$ or $O(kw^2 n \log(n))$

Table 1: Comparison among three sequence encoding layers with input sequence length n, network width w, kernel size k, and network depth D. Recurrent units and self-attention become slow as n grows. When the receptive field of a dilated convolution covers the longest possible sequence n, its overall computation is proportional to $O(\log n)$.

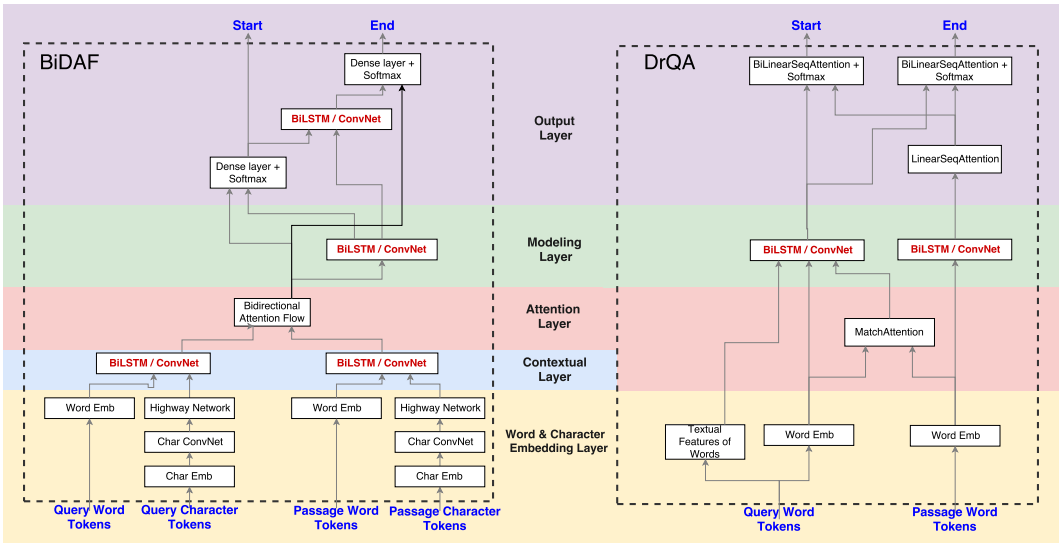

Figure 5: Schematic layouts of the BiDAF (*left*) and DrQA (*right*) architectures. We propose to replace all occurrences of BiLSTMs with diluted ConvNet structures.

of less than $n < 100$ words and a wide network is often used (i.e. d is large);
however, for reading comprehension tasks, where long documents and narrow networks are typical (i.e. $n \gg d$), self-attention becomes expensive. In addition, bidirectional recurrent units have the intrinsic problem that their sequential nature precludes parallel processing.

Admittedly, dilation has its limitations. It requires more overall computations than recurrent nets and the reception region is predetermined. We argue that to provide answers as a web service, one cares more about the response latency for a single question. Therefore, a short compute of the longest computation is favored.

1.3 Baseline Models: BiDAF and DrQA

Now we briefly describe two popular open-sourced question answering systems: Bi-directional Attention Flow (BiDAF) (Seo et al., 2017) and DrQA (Chen et al., 2017a), which are relevant to our study. Figure 5 shows schematic layouts of their respective model structures and highlight the BiLSTM layers (in red) which are the bottleneck for inference speed. Both models require LSTMs to encode the query and passage. BiDAF is more complex than DrQA, with two more LSTMs to make the classification decision.

The BiDAF model (Seo et al., 2017) introduces a bidirectional attention flow to help passing information between the passage and the query. It has six components: 1) character embedding layer, 2) word embedding layer, 3) contextual layer, 4) attention flow layer, 5) modeling layer, and 6) output layer, but only three of them contains LSTMs. The *contextual layer* encodes the passage and the query with two bidirectional LSTMs with shared weights. The *modeling layer* further employs a

two-layer stacked bidirectional LSTM to extract the higher order features of the words in the passage. The *output layer* uses yet another bidirectional layers to produces features for predicting the end of the answer span.

The DrQA system (Chen et al., 2017a) has a document retriever and a document reader. The *document retriever* simply uses pre-defined features to retrieve documents when the corresponding passage is not given in the question. The *document reader* uses two 3-layer stacked bidirectional LSTMs to encode the query and the passage/document respectively.

1.4 Related Work

Reading Comprehension Models. After the release of the Stanford Question Answering Dataset (Rajpurkar et al., 2016), reading comprehension models kept springing up in the past year. All of them use recurrent neural networks (Hochreiter & Schmidhuber, 1997; Cho et al., 2014) as a common component, and most of the top performed models uses attention (Bahdanau et al., 2014) in addition. RNet (Wang et al., 2017), demonstrates the effectiveness of the self-attention modules. Document Reader (DrQA) (Chen et al., 2017a) provides an question answering system using a document database. Hu et al. (2017) demonstrated how reinforcement learning can benefits the training procedure. SmartNet (Hermann et al., 2015) proposed to mechanism to keep refining the prediction.

ConvNets for Sequence Generation. There has been a lot of effort in applying ConvNet architectures to reduce the sequential computation in sequence to sequence models such as Extended Neural GPU(Kaiser & Bengio, 2016), ByteNet (Kalchbrenner et al., 2016) and ConvS2S Gehring et al. (2017). In these models, the number of operations required to relate signals from two arbitrary input or output positions grows with the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. The improvements in speed are limited to these generative models, because the decoding procedure still needs to be done token by token – and is therefore inherently linear with respect to the length of the to decoding sequence. In comparison, there is no generation in reading comprehension models (Seo et al., 2017; Chen et al., 2017a), and much more impressive speedups are possible through the application of ConvNets.

2 Model Specifics

The basic principle behind our approach is simple, yet very effective (as will be demonstrated in our experiments). We substitute the bidirectional sequence models with a simple convolutional network with repeated dilated convolutional layers. The receptive field of this convolutional network grows exponentially with depth and soon encompasses a long sequence, essentially enabling it to capture similar long-term dependencies as an actual sequential model. The compelling advantage of our approach is that the processing time is drastically reduced because convolution can be parallelized across the input passage. In this section we provide some details on our architecture. Figure 6 depicts a schematic layout of our proposed model, which we refer to as *Gated Linear Dilated Residual Network (GLDR).* It consists of two basic components, the dimensionality reduction and the residual block. In the following, we explain both in detail and then provide specifics about the application to BiDAF and DrQA.

The Dimensionality Reduction Block reduces the input (consisting of word embeddings and possibly other features) to a fixed dimensionality of 100 channels. It consists of a normal convolution (kernel size 3) and a gated linear unit (GLU) as activation and we train it with additional dropout regularization on the input (Hinton et al., 2012).

The Residual Block is the key ingredient to perform dilated convolution. It has a two-layer ConvNet with GLU activations (optionally with input dropout). The output of this small two-layer ConvNet is later summed up with the input allowing the ConvNet to learning only the residual of the transformation, similar to ResNet He et al. (2016). For simplicity all the convolutions are of kernel size 3 while the dilations vary across layers. To be more specific, the dilations of the convolutions in the first few residual blocks are increased exponentially $(1, 2, 4, 8, \cdots)$ with the purpose to increase the receptive field. After a small number of $O(\log(n))$ layers, the receptive field is wide enough and we switch back to normal convolutions for further refinement.

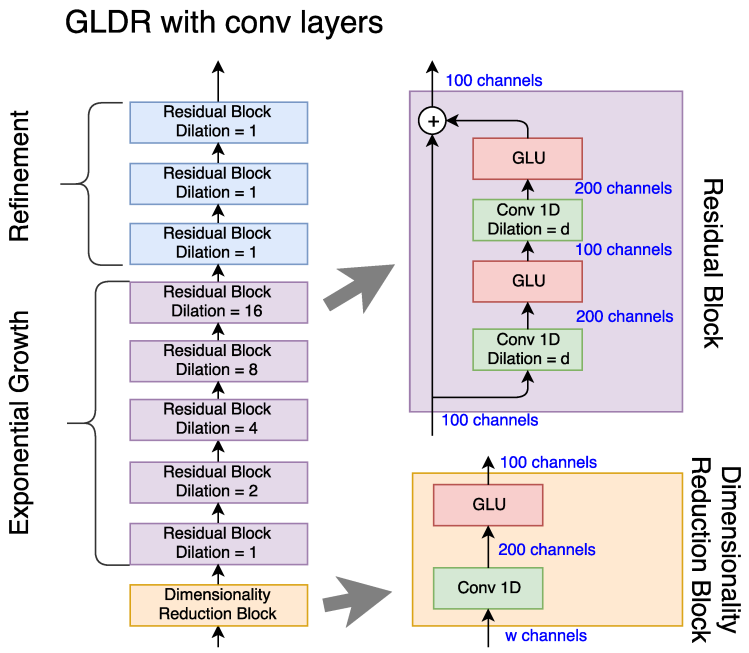

Figure 6: Schematic Layout of the Gated Linear Dilated Residual Network . The GLDR begins with dimensionality reduction to 200 channels through the use of a dimensionality reduction block (*bottom right*). Subsequently a short sequence of residual blocks (*right*) is used to increase the receptive field of the convolution exponentially. The output is further processed with a few layers of standard convolution (dilation 1).

To show ConvNets are not limited to a particular architecture of question answering models, we apply our GLDR to two popular open-sourced question answering systems: Bi-directional Attention Flow (BiDAF) (Seo et al., 2017) and DrQA (Chen et al., 2017a).

Convolutional BiDAF. In our convolutional version of BiDAF, we replaced all bidirectional LSTMs with GLDRs . We have two 5-layer GLDRs in the contextual layer whose weights are un-tied. In the modeling layer, a 17-layer GLDR with dilation 1, 2, 4, 8, 16 in the first 5 residual blocks is used, which results in a reception region of 65 words. A 3-layer GLDR replaces the bidirectional LSTM in the output layer. For simplicity, we use same-padding and kernel size 3 for all convolutions unless specified. The hidden size of all GLDRs is 100 which is the same as the LSTMs in BiDAF.

Convolutional DrQA. Since the query is much shorter than the document, we can afford using a 17-layer GLDR without dilation for encoding the query. However, a 9-layer GLDR whose 4 residual blocks have dilation 1, 2, 4, and 8, respectively is used to capture the context information in the passage, which results in a receptive region of 33 words. The hidden size of the ConvNet is 128 which matches that of the LSTMs in DrQA.

3 Experiments

We perform experiments on two data sets, the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017). In the following we describe the experimental setup, provide details on both data sets and elaborate on our experimental findings.

Experiment Setup. We evaluate convolutional versions of BiDAF and DrQA, which we refer to as Conv-BiDAF and Conv-DrQA respectively. We adopt the open-sourced BiDAF implementation[3] which is written in TensorFlow (Abadi et al., 2016) and DrQA implementation[4] in PyTorch [5], and follow their preprocessing and experimental setup. The models are trained with either a NVIDIA

[3] https://github.com/allenai/bi-att-flow
[4] https://github.com/facebookresearch/DrQA
[5] http://pytorch.org/

Tesla P100 GPU or a NVIDIA Titan X (Pascal) GPU, but the latter is used exclusively for all timing experiments. Across all experiments we only time GPU eclipsed time (i.e. forward and backward passes through the networks) since CPU bounded operations are not our focuses.

3.1 THE STANFORD QUESTION ANSWERING DATASET (SQUAD)

SQuAD is one of the most popular reading comprehension datasets and contains over 100K questions-answer-passage tuples. The data set was labeled by crowdsource workers who, given a passage, were asked to generate questions based on it. For each passage another group of workers attempted to highlight a span in the passage as the answer. This ensures that the passage also contains sufficient information and the answer is always present. We evaluate our experiments on the SQuAD validation set (which is publically available) as the secret test set is guarded with access limitations.

Experiment Setup For BiDAF and Conv BiDAF we try out a few optimization settings and picked the best one based on the validation set.

For the final BiDAF model, we use a batch size of 60, dropout rate 0.2 (Srivastava et al., 2014), and train it for 60000 iterations. In addition, we use stochastic gradient descend with momentum 0.1 and weight decay 10^{-4}, and decay the learning rate by a factor of 10 every 20000 iterations, which improves the performance of BiDAF slightly compared to training it with Adam (Kingma & Ba, 2014) for 20000 iterations suggested by Seo et al. (2017)[6].

For our Conv BiDAF, we train the model for 60000 iterations with the Adam optimizer (Kingma & Ba, 2014) using the default settings in TensorFlow ($\alpha = 0.0001, \beta_1 = 0.9, \beta_2 = 0.999$), drop α by a factor of 10 every 20000 iterations, and use an additional word dropout rate 0.1 (Dai & Le, 2015). Word dropout isn't found helpful for BiDAF in our experiment. Because of the GPU memory constraint, the model is trained with the documents shorter than or equal to 400 word tokens as what the authors did in the paper.

For all DrQA variants, we adopt batch size 32, dropout rate 0.3, and train both models for 60 epochs with Adamax (Kingma & Ba, 2014) optimizer using the default setting in PyTorch ($\alpha = 0.002, \beta_1 = 0.9, \beta_2 = 0.999$). Weight decay and word dropout doesn't result in a fair amount of improvement on either of the models, so they are abandoned in the reported models. The models are trained on the SQuAD training set without removing long documents.

Results. Figure 8 shows the F1 score on the development partition of the SQuAD dataset for the various algorithms, Conv BiDAF, Conv DrQA, BiDAF, DrQA, and R-NET (Wang et al., 2017) as a function of inference GPU time. The figure shows the results across all documents (middle) and for the top 10% shortest and longest documents (left and right plot respectively). Throughout, Conv-BiDAF achieves one to two order of magnitude speed-up at inference time and performs almost as well as the original BiDAF.The plot also shows the F1-score of the R-Net model for which we have no inference timing (therefore shown as a horizontal line). More detailed comparisons between BiDAF and Conv-BiDAF are shown in Table 2.

On Table 3, we show different variants of BiDAF models and their performance.

Model # of params	BiDAF 2.70M	Conv BiDAF (5-17-3) 2.76M
Dev EM	67.66 ± 0.41	68.28 ± 0.56
Dev F1	77.29 ± 0.38	77.27 ± 0.41
Training GPU time (h) until convergence	21.5[7]	3.4 (6.3x)
Training GPU time (sec) per iteration, batch size = 60	3.88 ± 2.14	0.20 ± 0.14 (19x)
Inference GPU time (sec) per iteration, batch size = 60	1.74 ± 0.75	0.081 ± 0.002 (21x)
Inference GPU time (sec) per iteration, batch size = 1	1.58 ± 0.06	0.016 ± 0.003 (98x)

Table 2: BiDAF v.s. Conv BiDAF. EM stands for exact match score.

[6]Based on `https://github.com/allenai/bi-att-flow/issues/10`, the authors switch from Adadelta to Adam.

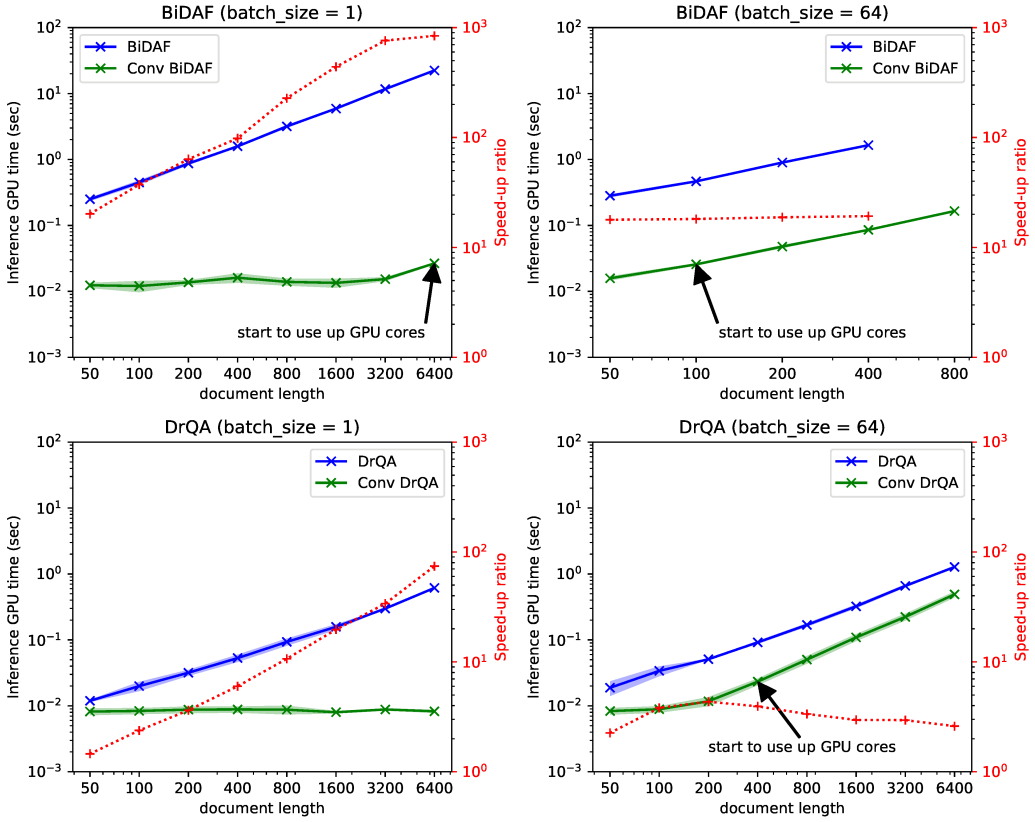

Figure 7: Inference GPU time of four models with batch size 1 or 64. The time spent on data pre-processing and decoding on CPUs are not included. The lines of Convolutional models should be almost flat as long as their are enough cores in the GPU. The DrQA uses the CuDNN LSTM implementation which makes it much faster than BiDAF. The missing points for BiDAF and Conv-BiDAF were caused by running of out GPU memories. We use a single NVIDIA Titan X (Pascal) with 12GB memory is this experiment.

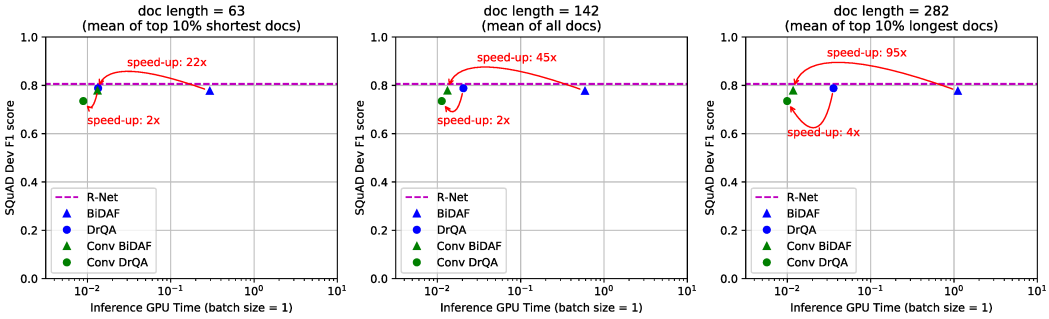

Figure 8: Dev F1 Score on SQuAD vs Inference GPU Time for a variety of doc lengths. Our proposed methods, Conv BiDAF and Conv DrQA, approximately maintain the high accuracies of the original models while introducing tremendous speedups at inference time.

[7]To make it fair we report the training time of BiDAF for 20000 iterations. Both our BiDAF and Conv-BiDAF were trained for 60000 iterations to reach the best dev F1 based on hyperparameter tuning, but Seo et al. (2017) suggests training it for only 20000 iterations which achieves a sightly worse results for BiDAF.

Model	# of params	Dev EM	Dev F1
BiDAF (trained by us)	2.70M	67.94	77.65
Conv BiDAF (5-17-3 conv layers)	2.76M	68.87	77.76
Conv BiDAF (9-9-3 conv layers)	2.76M	67.79	77.11
Conv BiDAF (0-31-3 conv layers)	3.36M	63.52	72.39
Conv BiDAF (11-51-3 conv layers)	5.53M	**69.49**	**78.15**
Conv BiDAF (31-31-3 conv layers)	6.73M	68.69	77.61
DrQA (trained by us)	33.82M	**69.85**	**78.96**
Conv DrQA (9 conv layers)	32.95M	62.65	73.35

Table 3: Comparing variants with different number of layers. EM stands for exact match score. The scores are multiplied by 100. DrQA uses a much larger pre-trained word embedding resulting in more parameters.

Model	Dev EM	Dev F1
Conv BiDAF (5-17-3 conv layers)	**68.87**	**77.76**
without dilation	68.15	76.99
Using ReLU instead of GLU	63.91	73.39

Table 4: Ablation Test. EM stands for exact match score. The scores are multiplied by 100. Without residual learning (shortcut connection) the model cannot learning meaningful information and diverges.

Table 4 compares various structural choices for the convolutional architecture. BiDAF(11-51-3) produces the best answer quality, but the model is also larger, and therefore more expensive for inference. Overall, all ConvNet models are much smaller than the LSTM based models, while still able to produce comparable answer quality.

Table 3 gives the result of ablation studies. We can see that dilation helps improve answer quality a bit. The choice of non-linear unit has a huge impact on answer quality with GLU performs much better than ReLU. Without residual learning (removing the shortcut connection), the model can never converge to fair point.

3.2 TRIVIAQA

TriviaQA is a large-scale reading comprehension dataset with 95K question-answer pairs and 650K question-answer-evidence tuples which is more challenging than SQuAD because it 1) contains more complex questions, 2) has substantial syntactic and lexical variability in the text, 3) requires a significant amount of cross-sentence reasoning, and 4) the answer and the sufficient information are guaranteed in the evidence. For each question-answer pair, it used distant supervision to provide relevant evidence from wikipedia or web search. Besides the full development and test set. A verified subset for each is also provided.

With the same hyperparamters used for SQuAD, our Conv DrQA outperforms all models reported in the published literatures while being slightly worse than the unpublished ones on the wiki split leader-board. Again, we can see a trade-off between performance and speed.

Experiment Setup We process the data into SQuAD format with the script provided by Trivia QA[8]. Precisely, for each document in the candidate set of a question-answer pair, it produces a question-answer-evidence tuple for training as long as any of the answers appear in the first 800 tokens in the document. For evaluation, we truncate each document down to 1600 tokens and predict a span among them. We follow the suggestion from Joshi et al. (2017) to use only the first 80K question-answer-evidence tuples (out of 529K) of the Web split of TriviaQA for training.

Results. On Wikipedia split of TriviaQA, our proposed Conv DrQA is slight worse than our DrQA baseline which beats all previous models, it can still be on a par with the previous state-of-the-art performance of recurrent networks. The numbers are shown in Table 5 The Conv DrQA model only encode every 33 tokens in the passage, which shows that such a small context is enough most of the question.

[8]https://github.com/mandarjoshi90/triviaqa

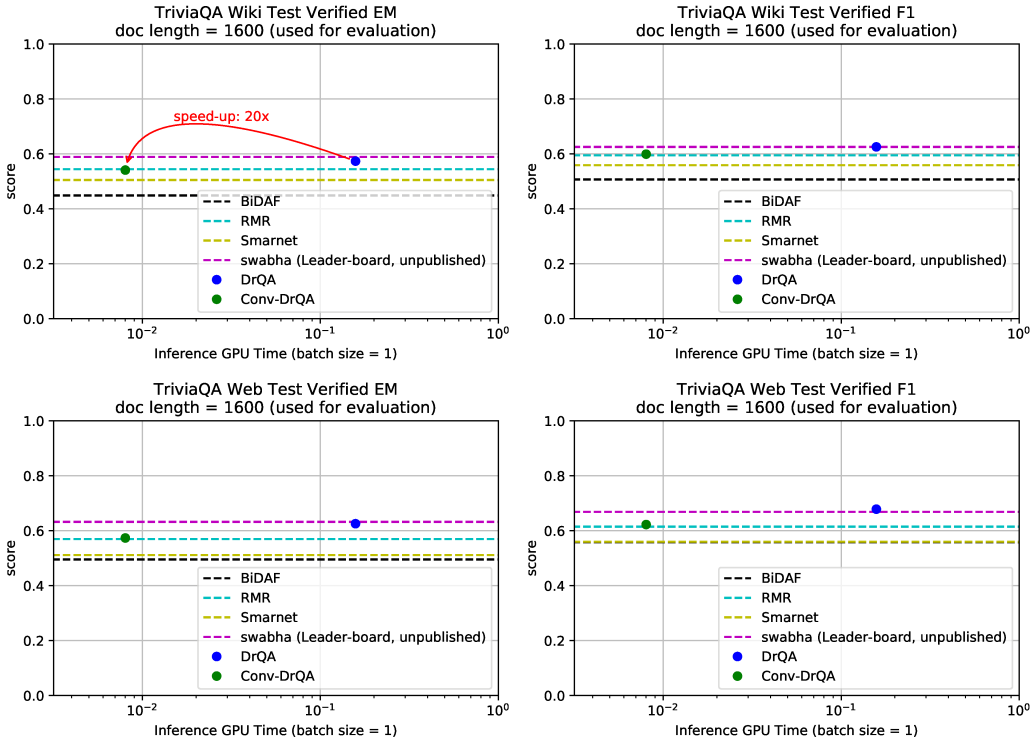

Figure 9: Test exact match and F1 Score on Wikipeida and web split of TriviaQA vs Inference GPU Time for the maximum document length used in our evaluation. Our proposed methods, Conv BiDAF and Conv DrQA, approximately maintain the high accuracies of the original models while introducing tremendous speedups at inference time.

3.3 Memory usage

Figure 10 compares the memory usage of different types of models. We can see that the quadratic growth of self-attention may prevents its use with most real-world documents. The experiment is done as follows. We fix input size and the hidden size of the layers to be 100 and use the PyTorch implementation. The batch size is fixed to 64. For self-attention the attention matrix is computed by forwarding the input vectors through a dense layer with ReLU activation and then taking the inner product between any pairs of two outputs of this dense layer.

4 Conclusion

We propose a convolutional architecture as an alternative to the recurrent architectures typically used in reading comprehension models. By using simple dilated convolutional units in place of recurrent models, we achieve results comparable to the state of the art on two question answering tasks, while at the same time achieving up to two orders of magnitude speedups at inference time. Most applications of question answering (e.g. search engines or mobile assistant) are particularly sensitive to latency and even a small speedup can be a huge improvement.

Our results raise the question for which other tasks in NLP sequence models are not necessary and can be replaced by dilated convolution. If the purpose of the sequence model is to enable long-range dependencies, it may be that convolution with very large receptive fields could be sufficient. In this paper, we provide evidence that in the case of reading comprehension this may be the case.

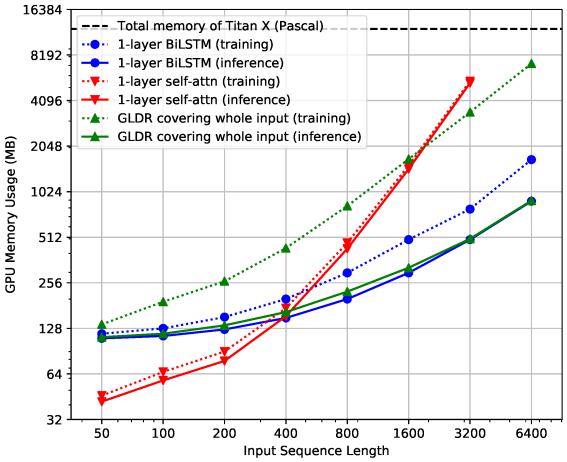

Figure 10: GPU memory usage of a single layer self-attention versus a GLDR with enough depth whose receptive field covers the whole sequence. The number of convolution layers of GLDR is 15 when the input sequence length is 50, and increases by 2 (adding one more residual block) each time the sequence length doubles. The missing points are caused by out of GPU memory.

ACKNOWLEDGMENTS

We are grateful to Avinash Atreya, Neil Houlsby, Tom Kwiatkowski, and Lukasz Kaiser for helpful discussions during the course of this work.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Petr Baudiš and Jan Šedivý. Modeling of the question answering task in the yodaqa system. In *Proceedings of the 6th International Conference on Experimental IR Meets Multilinguality, Multimodality, and Interaction - Volume 9283*, CLEF'15, pp. 222–228, New York, NY, USA, 2015. Springer-Verlag New York, Inc. ISBN 978-3-319-24026-8. doi: 10.1007/978-3-319-24027-5_20. URL http://dx.doi.org/10.1007/978-3-319-24027-5_20.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1533–1544, 2013.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*, 2017a.

Zheqian Chen, Rongqin Yang, Bin Cao, Zhou Zhao, Deng Cai, and Xiaofei He. Smarnet: Teaching machines to read and comprehend like human. 2017b.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1179.

Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. *https://arxiv.org/abs/1710.10723*, 2017.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pp. 3079–3087, 2015.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 933–941, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1243–1252, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778. IEEE Computer Society, 2016.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

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

Minghao Hu, Yuxing Peng, and Xipeng Qiu. Reinforced mnemonic reader for machine comprehension. 2017.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Lukasz Kaiser and Samy Bengio. Can active memory replace attention? In *NIPS*, pp. 3774–3782, 2016.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *CoRR*, abs/1610.10099, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 1400–1409, 2016.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *EMNLP*, pp. 2249–2255. The Association for Computational Linguistics, 2016.

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

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *ICLR*, 2017.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

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

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

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

A MORE EXPERIMENTS

A.1 TRIVIAQA SCORES

We report the full and verified exact-match and F1 scores in Table 5

Dataset	Model	Full		Verified	
		EM	F1	EM	F1
Wikipedia	BiDAF (Joshi et al., 2017)	40.32	45.91	44.86	50.71
	RMR (Hu et al., 2017)	46.94	52.85	54.45	59.46
	Smarnet (Chen et al., 2017b)	42.41	48.84	50.51	55.90
	swabha_ankur_tom (unpublished) (Leader-board [8])	51.59	55.95	58.90	62.53
	DocQA (Clark & Gardner, 2017)	-	-	-	-
	DrQA	52.58	58.17	57.36	62.55
	Conv DrQA	49.01	54.52	54.11	59.90
Web	BiDAF (Joshi et al., 2017)	40.74	47.06	49.54	55.80
	RMR (Hu et al., 2017)	46.65	52.89	56.96	61.48
	Smarnet (Chen et al., 2017b)	40.87	47.09	51.11	55.98
	swabha_ankur_tom (unpublished)	53.75	58.57	63.20	66.88
	DocQA (Clark & Gardner, 2017) (Leader-board [8])	66.37*	71.32*	79.97*	83.70*
	DrQA	51.49	57.87	62.55	67.84
	Conv DrQA	47.77	54.33	57.35	62.23

Table 5: TriviaQA Performance. The scores are multiplied by 100. *Doc QA (Clark & Gardner, 2017) uses a different training strategy on the whole training set (containing 529K question-answer-evidence tuples), while we follow Joshi et al. (2017) training on only the first 80K question-answer-evidence tuples in the training set.

A.2 OPEN-DOMAIN QUESTION ANSWERING

Following Chen et al. (2017a), we further evaluate the full Conv DrQA system which uses the document retriever in DrQA to filter the documents and answers open-domain questions. We test the model on three open-domain query sets – CurateTREC Baudiš & Šedivý (2015), WebQuestions Berant et al. (2013), and WikiMovies Miller et al. (2016). Table 6 compares these datasets. We show that Conv DrQA achieves comparable results as DrQA on these open-domain questions.

Experiment Setup On open-domain datasets, only question-answer pairs are provided. We use the Wikipedia pages provided by Chen et al. (2017a) which contains 5 million pages to provide distant supervision. To be more specific, given a question-answer pair, we find the Wikipedia pages containing the answer and treat it as an evidence to generate a question-answer-evidence tuple. However, this distant supervision can only be used for generating training data. To evaluate the model, we have use the document retriever to retrieve relevant documents. The document reader rank all the Wikipedia pages using TF-IDF and bigram-hash. The document reader in DrQA or Conv DrQA finds an answer span in the top 5 documents. We start with a document reader pre-trained on SQuAD and fine-tune it with distant supervision as proposed by Chen et al. (2017a). We randomly sample 20% of the question-answer-evidence tuples in the original training set as the

[8] The best scores recorded on Nov 9, 2017 from the Leader-board `https://competitions.codalab.org/competitions/17208#results`.

Dataset	# of questions		# of tuples for Distant Supervision	
	Full Train	Test	Train	Dev
CuratedTrec	1,486	694	2,935	735
WebQuestions	3,778	2,032	5,736	1,464
WikiMovies	96,185	9,952	75,174	18,752

Table 6: Statistics of the open-domain datasets.

	CuratedTrec	WebQuestions	WikiMovies
DrQA Chen et al. (2017a)	25.7	19.5	34.3
DrQA (reproduced by us)	25.9	**20.1**	35.5
Conv DrQA	**27.2**	19.1	**36.2**

Table 7: Fine-tuning results on open-domain datasets. All the scores are the exact match scores on the test set and multiplied by 100.

development set. We tune the the learning rate and the dropout rate, and select our model based on the exact match score on this development set.

We use the DrQA source code to generate the distant supervision data; however, we observe different statistics from what was reported by Chen et al. (2017a). The statistics of the generated distant supervision data is shown in Table 6.

Results Table 7 shows the test exact match of the fine-tuning models. Our Conv DrQA outperforms DrQA on the CuratedTrec and the WikiMovies datasets, while being a little worse on the WebQuestions dataset.