QANet with Universal Transformer

Kongphop Wongpattananukul
Department of Energy Resources Engineering
Stanford University
Stanford, CA 94305
kongw@stanford.edu

Abstract

Transformer encoder and decoder architecture [Vaswani et al., 2017] that compose of only self-attention and feed-forward recently gain a lot of traction recently due to its outstanding result especially in machine translation. QANet architecture [Yu et al., 2018], one of many application inspired by Transformer encoder, is built to tackle a reading comprehension task, a question answering on a given context, by replacing all recurrent neural network (RNN) which is a standard design for a reading comprehension task such as BiDAF from Seo et al. [2016] to use only convolution and self-attention for local and global interaction respectively. This architecture show an impressive performance by achieving state-of-the-art EM/F1 score in SQuAD 1.1 and speed-up training and testing time significantly over model with RNN. However, dropping of RNN in QANet also discard its advantage which is an inductive bias from iterative learning that is crucial for language understanding task [Dehghani et al., 2018]. We propose QANet with Universal Transformer by introducing recurrent transformation through depth that used shared parameters and timestep encoding from Universal Transformer [Dehghani et al., 2018]. This model achieves EM/F1 score of 62.4/66.3 on SQuAD 2.0 test set in non-PCE leaderboard and proves that recurrent transformation is useful in reading comprehension task.

1 Introduction

Due to recent invention of Transformer architecture [Vaswani et al., 2017] that process sequence data by solely relied on attention mechanism, many applications in natural language processing (NLP) domain start to use Transformer as an alternative building block to RNN because of its parallelizable nature. Similarly, an end-to-end reading comprehension system like BiDAF [Seo et al., 2016] that traditionally compose of RNN and context-query attention also inspired QANet [Yu et al., 2018] that embraced Transformer-based encoder in place of RNN. Although QANet outperform RNN architecture and achieve state-of-the-art in SQuAD 1.1, Dehghani et al. [2018] argued that RNN's inductive bias is still useful and crucial to language understanding task. Dehghani et al. [2018] introduced Universal Transformer that introduced recursive transformation through depth to retain RNN's inductive bias feature.

In this paper, we introduce QANet with Universal Transformer that incorporate Universal Transformer [Dehghani et al., 2018] idea on recurrent transformation to a Transformer-based reading comprehension system. We still retain fundamental structure of QANet [Yu et al., 2018] system that consist of embedding, embedding encoder, context-query attention, model encoder and output layer but change encoder block from Transformer to Universal Transformer. The integration of idea from Universal Transformer is very simple. Each convolution and feed-forward layer in encoder block is switched to weight sharing across the encoder stack with time encoding to tag the recursive operation. Universal Transformer utilize recursion through depth to refine representation of each position instead of recursion through sequence as RNN [Dehghani et al., 2018].

Our model outperforms QANet with similar architecture on SQuAD 2.0 development set and get a competitive result (EM/F1 score of 62.4/66.3) on SQuAD 2.0 test set in non-PCE leaderboard. We also provide error analysis on why our model get incorrect prediction and possible solution to tackle those issue using additional dependency feature or long-term dependency.

2 Related Work

A reading comprehension task, a question answering on a given context, recently gain a lot of popularity in NLP research domain due to publicly available dataset such as SQuAD 2.0 [Rajpurkar et al., 2016]. One of the recent end-to-end reading comprehension system is BiDAF from Seo et al. [2016] that introduced a novel attention mechanism which is bi-directional (contrast to traditional context-query attention mechanism that is only uni-directional). The model compute representation from character-level [Kim, 2014], word-level [Pennington et al., 2014] and contextual embedding then bridge context-query with bi-direction attention. Subsequently, the summarization is passed through RNN to scan the context for output. On the other hand, Transformer architecture that based solely on self-attention mechanism is introduced by Vaswani et al. [2017] to tackle machine translation task. For encoder-decoder architecture, it can directly replace RNN, a tradition building block on sequence model, to reduce sequential computation and make model more parallelizable. QANet [Yu et al., 2018] is a product of this idea on reading comprehension task that replace all RNN encoder in BiDAF with Transformer-based encoder to achieve state-of-the-art performance in SQuAD 1.1. Additionally, Yu et al. [2018] also introduce data augmentation technique for reading comprehension task using neural machine translation (NMT) to enhance model performance. Recently, Universal Transformer [Dehghani et al., 2018] further improve Transformer architecture by introducing recurrent structure through depth and time encoding to retain inductive bias from RNN architecture that proven to be useful for language understanding. This idea could further improve system that based on Transformer like QANet as we show in QANet with Universal Transformer.

3 Approach

In this section, we describe the structure of our proposed model which is QANet with Universal Transformer. We adopt QANet framework [Yu et al., 2018] that is proven to be very effective for reading comprehension task using only convolution, self-attention and feed-forward mechanism and change its encoder building block that consist of convolution, self-attention and feed-forward layer to self-attention and recurrent transformation layers introduced in Universal Transformer [Dehghani et al., 2018].

3.1 Problem Definition

The SQuAD challenge [Rajpurkar et al., 2016] which is a reading comprehension task consist of context/paragraph ($C = \{c_1, c_2, ..., c_m\}$) with m words, question/query with n words ($q = \{q_1, q_2, ..., q_n\}$) and answer with $p+1$ words chosen from context ($A = \{c_i, c_{i+1}, ..., c_{i+p}\}$). Given context and query, we want to propose an answer that span in the given context.

3.2 Model Overview

QANet with Universal Transformer composes of 5 major components (embedding, embedding encoder, context-query attention, model encoder and output layer) as show in Figure 1 same as QANet. We modify our encoder block to encompass recurrent transformation from timestep encoding and shared parameters introduced in Universal Transformer [Dehghani et al., 2018].

3.2.1 Input Embedding Layer

We adopt embedding layer used in BiDAF [Seo et al., 2016] and QANet [Yu et al., 2018] that utilize both word embedding and character embedding. Word embedding maps each word from input context/query to word vectors ($\mathbb{R}^{300}$) with GloVe [Pennington et al., 2014] pre-trained word vectors. Character embedding obtains its vector ($\mathbb{R}^{200}$) from convolutional neural network (CNN) of mapped character vector ($\mathbb{R}^{64}$) from random initialization proposed by Kim [2014] (kernel size is 5 and number of filter is 200 with maxpooling layer afterward). Then, word embedding and character

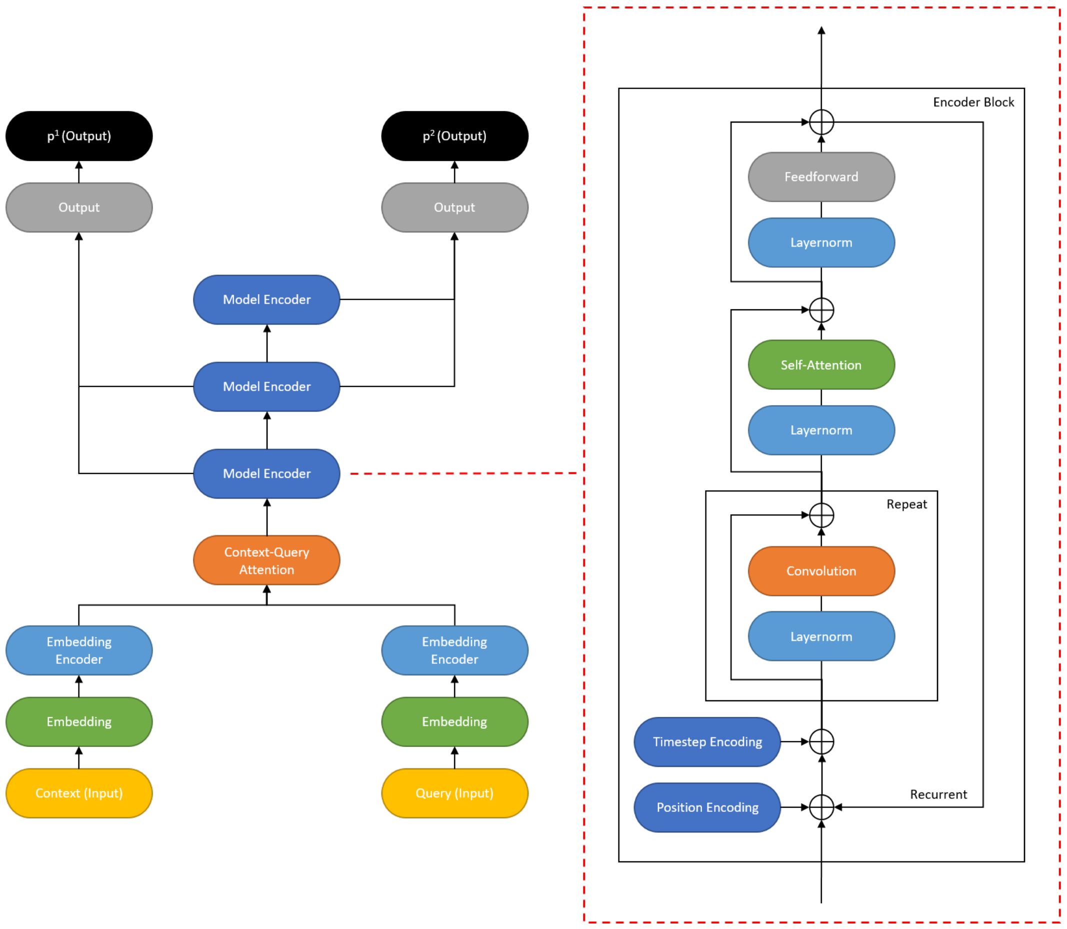

Figure 1: QANet with Universal Transformer architecture and its encoder block

embedding are concatenated ($\mathbb{R}^{300+200}$) and pass through linear projection layer to reduce dimension from $\mathbb{R}^{500}$ to $\mathbb{R}^{d}$ where $d = 128$ is a numbers of hidden size. Finally, Two-Layer Highway Network [Srivastava et al., 2015] is applied to get output of this layer ($\mathbb{R}^{m \times d}$). This layer is applied to context and query separately to get each embedding vectors individually but their weights are shared to get similar contextual embedding.

3.2.2 Embedding Encoder Layer

We also utilize QANet encoder block structure which have convolution layer (depthwise separable convolution from Chollet [2016]; kernel size is 7 and number of filter is $d = 128$, number of convolution layers is 4), self-attention layer (multi-head attention as shown in Equation 1 from Vaswani et al. [2017]; number of heads is 4, similarity score is scaled dot product) and feed-forward layer (position-wise feed-forward network from Vaswani et al. [2017]) as basic building block placed inside residual block and layer-normalization ($f(layernorm(x)) + x$).

$$MultiHead(Q, K, V) = W^{O}[head_1, ..., head_4] \tag{1}$$

$$head_i = Attention(W_i^{Q}Q, W_i^{K}K, W_i^{V}V) \tag{2}$$

$$Attention(Q, K, V) = softmax(\frac{QK^{T}}{\sqrt{d_k}})V \tag{3}$$

where Q, K, V are query, key and value, $W^{O}, W_i^{Q}, W_i^{K}, W_i^{V}$ is trainable parameter in linear projection and d_k is the dimension of query and key.

However, we used *shared* convolution layer and *shared* feed-forward layer with timestep encoding [Dehghani et al., 2018] as shown in Figure 1 instead of stacking new encoder block together to build a deeper layer. P^t is the two-dimensional coordinate embedding that combined both position and time encoding (where $\oplus$ is an element-wise summation).

$$P^t_{pos,2j} = \sin(pos/10000^{2j/d}) \oplus \sin(t/10000^{2j/d}) \tag{4}$$

$$P^t_{pos,2j+1} = \cos(pos/10000^{2j/d}) \oplus \cos(t/10000^{2j/d}) \tag{5}$$

The number of recurrent is 1 similar to 1 encoder block in QANet and their weights are shared between context and query embedding encoder layer.

3.2.3 Context-Query Attention Layer

In order to build context and query interaction, we employ context-query attention from QANet [Yu et al., 2018] which is a layer to combine context ($C \in \mathbb{R}^{m \times d}$) and query ($Q \in \mathbb{R}^{n \times d}$) in reading comprehension task that compose of both context-to-query attention ($\tilde{Q} \in \mathbb{R}^{m \times d}$) and query-to-context attention ($\tilde{C} \in \mathbb{R}^{d \times n}$) introduced by Seo et al. [2016]. We compute the similarity matrix ($S \in \mathbb{R}^{m \times n}$) using trilinear function:

$$S_{ij} = W_0[q_j, c_i, q_j \odot c_i] \tag{6}$$

where W_0 is a trainable parameter.

For context-to-query attention, we find attention weight by applying softmax function to each row in similarity matrix ($\overline{S}$) then compute attention from dot product of attention weight and its corresponding query ($\tilde{Q} = \overline{S}Q$). For query-to-context attention, we compute attention as $\tilde{C} = \overline{S}\,\overline{\overline{S}}^T C$ where $\overline{\overline{S}}$ is column-wise softmax normalization of similarity matrix. The combined context and query output is the concatenation of context (C), context-to-query attention ($\tilde{Q}$) and query-to-context attention ($\tilde{C}$) as $[c, \tilde{q}, c \odot \tilde{q}, c \odot \tilde{c}] \in \mathbb{R}^{m \times 4d}$

3.2.4 Model Encoder Layer

The input from context-query attention layer is immediately mapped from $\mathbb{R}^{m \times 4d}$ to $\mathbb{R}^{m \times d}$ using linear projection. Then, we employed similar encoder block as introduce in embedding encoder layer except that we change the number of convolution layer to 2 with kernel size of 5 and the number of recurrent to 7. In addition, we stack 3 identical model encoder (shared weights) on top of each other to compute multiple model encoder output ($M \in \mathbb{R}^{m \times d}$).

3.2.5 Output Layer

For output layer that we need to identify answer within the context (A) for SQuAD challenge, we adopt the strategy to compute probability of starting point (p^1) and end point (p^2) used by Seo et al. [2016] and Yu et al. [2018]. To compute the probability, we perform linear projection on concatenated of model encoder output then normalize with softmax function.

$$p^1 = softmax(W_1[M_0; M_1]) \tag{7}$$

$$p^2 = softmax(W_2[M_1; M_2]) \tag{8}$$

where M_0, M_1, M_2 are three output from successive model encoder and W_1, W_2 are trainable parameter in linear projection.

The answer score is the product of these two probabilities. For optimization, we defined loss as negative sum of log probability of labeled start and end position in context that average across all training examples.

Test

For test time, the answer is chosen as span (a, b) in context such that the value of $p^1_a p^2_b$ is maximized where a, b is the indices of word in context with $a < b$.

Contribution

From baseline model, we implement CharCNN [Kim, 2014] to get baseline with CharCNN. Then, we implement QANet encoder block and output layer using some component from Transformer implementation[1]. Finally, we modify QANet encoder block to QANet with Universal Transformer encoder block using time encoding and recurrent structure.

4 Experiments

4.1 Data

We use the Stanford Question Answering Dataset (SQuAD) [Rajpurkar et al., 2016] which is a reading comprehension dataset curated from Wikipedia article. The latest version of SQuAD is 2.0 that also include question with no answer. Although Yu et al. [2018] introduced a novel data augmentation on SQuAD 1.1 using neural machine translation (NMT) to paraphrase the context and its corresponding answer to a given question, we limit ourselves only to a given SQuAD 2.0 without any augmentation and focus on building a model that could generalized well to unseen data. SQuAD 2.0 contains 141.9K context-query pairs with 129.9K for training, 6.1K for development and 5.9K for testing. Similar to Yu et al. [2018], context and question is tokenized and limited to 400 and 50 maximum words respectively where CharCNN only use 16 maximum characters for each word. The maximum answer length is set to 30.

4.2 Evaluation Method

The SQuAD challenge used Exact Match (EM) and F1 score to determine how well our prediction fare with ground truth from multiple human responses. Exact match (EM) is a binary score (0/1) comparison of prediction with ground truth which is very strict criteria compare to F1 that is a harmonic mean of precision and recall between prediction and ground truth. The total EM/F1 score are averaged across all evaluation data to get overall performance.

4.3 Experimental Details

For QANet and QANet with Universal Transformer, we adopt hyperparameter from Yu et al. [2018] with minor modification.

- Optimizer: ADAM [Kingma and Ba, 2014] with constant learning rate of 0.001 and $\beta_1 = 0.8, \beta_2 = 0.999, \epsilon = 10^{-7}$
- Regularization: Word and character dropout probability of 0.1 and 0.05 respectively, stochastic depth method (layer dropout) [Huang et al., 2016] in embedding encoder layer and model encoder layer with last layer survival probability of 0.9 and L2 weight decay with $\lambda = 3 \times 10^{-7}$
- Variable: Exponential moving average with a decay rate of 0.9999
- Hidden Size: 128
- Batch Size: 32

4.4 Results

Table 1 show the performance of model on SQuAD 2.0 that improve over baseline model using CharCNN, QANet and Universal Transformer as well as published result of QANet [Yu et al., 2018] on SQuAD 1.1 for benchmarking.

At first, character embedding raise model performance significantly over baseline (+2.4/+2.2 for EM/FM score) as expected because it could build word representation from character input which is very useful for out-of-vocabulary word. Then, we adopt QANet architecture with hidden size of 80 contrast to hidden size of 128 due to limitation of GPU memory. We benchmark our implementation on question with answer in SQuAD 2.0 with published result from Yu et al. [2018] on SQuAD 1.1 as shown in Table 1. Our implementation get a bit lower EM/F1 score because we implement a simpler version of QANet (lower hidden size and lower character vector size). In addition, we also train our model on SQuAD 2.0 that might also hinder its performance on question with answer (It also need to balance between question with answer and question without answer). Nevertheless, its performance

[1]`http://nlp.seas.harvard.edu/2018/04/03/attention.html`

is inline with the published one. Finally, QANet with Universal Transformer further enhance our model capability from its recursive transformation on top of QANet by +0.8/+1.1 (EM/F1 score) with same architecture and +2.8/+2.9 (EM/F1 score) for a larger one (capable because of shared parameters in Universal Transformer). Comparing the objective function which is a negative log likelihood (NLL) between training set and development set show a bit of overfitting problem that is regulated by our regularization scheme (L2 weight decay and dropout).

The best model achieve EM score of 62.4 and F1 score of 66.3 on SQuAD 2.0 test set in non-PCE leaderboard.

Table 1: The performance of models on SQuAD 2.0

		Q w/ A		Overall	
Model	Dataset	EM	F1	EM	F1
Baseline	Dev set	68.9	77.9	58.4	61.7
Baseline with CharCNN	Dev set	69.6	77.8	60.8	63.9
QANet ($d = 80, head = 4$)	Dev set	69.5	78.5	63.2	66.7
QANet with UT ($d = 80, head = 4$)	Dev set	67.9	78.3	64.0	67.8
QANet with UT ($d = 128, head = 4$)	Dev set	67.5	77.0	**66.0**	**69.6**
QANet with UT ($d = 128, head = 4$)	Test set	-	-	**62.4**	**66.3**
QANet [Yu et al., 2018]	Dev set	**73.6**	**82.7**	-	-

5 Analysis

In order to understand why our model get incorrect prediction, we randomly select 100 predictions and 100 errors from QANet with Universal Transformer ($d = 128, head = 4$) model on SQuAD 2.0 development set and manually evaluate the result individually. Table 2 show the number of result in each category for both question with answer (Q w/ A) and question without answer (Q w/o A) using EM/F1 score criteria. From prediction, we observed that the model is doing a pretty good job at identifying question with answer (45/50 examples) while struggle a bit for question without answer (39/50 examples). However, the model prediction only got half of them perfectly right in question with answer (26/50 examples) which is lower than 39/50 examples for question without answer. Still, it is difficult to evaluate performance of model solely on EM/F1 score as shown in Table 1 and Table 2 because there are many different reason why the model get incorrect prediction.

Table 2: Prediction of QANet with Universal Transformer on SQuAD 2.0 development set

	# of Data	
Result	Q w/ A	Q w/o A
Match		
Exact Match	26	39
Partially Match	14	-
Not Match	5	11
No Answer	5	-
Total	50	50

Further investigation on the error prediction (partially match, not match and no answer) reveal some insight toward model performance as shown in Table 3 with correct (C) and incorrect (I) answer definition. The inconclusive label is defined for examples that we are not certain in our interpretation between prediction and labeled answer.

For question with answer, around two-thirds of partially match category are acceptable answers that doesn't fit with EM/F1 score criteria. These answers usually contain additional context (e.g. noun phrase, preposition phrase) or only missing minor detail (e.g. article) which should be able to resolve by computing dependency structure before evaluate with F1 score. The rest of partially

Table 3: Error analysis of QANet with Universal Transformer on SQuAD 2.0 development set

	# of Data	
Type of Error	Q w/ A	Q w/o A
Partially Match	(28)	(-)
(C) Additional context	9	-
(C) Missing irrelevant context	11	-
(I) Dependency structure	3	-
(I) Missing relevant context	5	-
Not Match	(15)	(37)
(I) Misunderstand question	4	-
(I) Matching to relevant context	-	4
(I) Matching to irrelevant context	9	31
(I) Inconclusive	2	2
No Answer	(20)	(-)
(I) Coreference	10	-
(I) Inconclusive	10	-
Total	63	37

match is incorrect due to dependency structure (e.g. missing preposition phase in answer or confuse with dependency structure in context). The other big issue is no answer prediction which half of them come from coreference in multiple sentence (e.g. usage of pronoun, name entity) as shown in example below. The "western empires" in question refer to both "French and British" and "imperial and colonial" in context and the model need to understand its relationship to generate a correct answer. It seem like the model could not located information through coreference in context and rely on concise and self-contained sentence to answer the question. Lastly, the answer that doesn't match with human response commonly misinterpret either question or context and output incorrect prediction (e.g. incorrect dependency of preposition phrase, noun phrase).

Context:	To better illustrate this idea, Bassett focuses his analysis of the role of nineteenth-century maps during the "scramble for Africa". He states that maps "contributed to empire by promoting, assisting, and legitimizing the extension of **French and British** power into West Africa". During his analysis of nineteenth-century cartographic techniques, he highlights the use of blank space to denote unknown or unexplored territory. This provided incentives for **imperial and colonial** powers to obtain "information to fill in blank spaces on contemporary maps".
Question:	What provided an incentive to **western empires** to colonize Africa?
Answer:	blank spaces on contemporary maps
Prediction:	N/A

In contrast, question without answer have a lot of issue with irrelevant context. The large portion of these error come from the fact that model misunderstand noun phrase in question and context to be the same and generate answer accordingly. For instance, the question ask for "rare autoimmune disease" but the model answer with "autoimmune diseases" instead. Interestingly, some question without answer contain a context pattern that commonly contain answer in question with answer (e.g. parallel of 2 consecutive sentences in different language for translation, define a meaning of specific word) and successfully mislead the model to generate incorrect answer.

Context:	Disorders of the immune system can result in **autoimmune diseases**, inflammatory diseases and cancer.
Question:	What is a **rare autoimmune disease?**
Answer:	N/A
Prediction:	inflammatory diseases and cancer

In summary, most of the error in both question with answer and question without answer are boiled down to the problem of dependency structure. The model incorrectly identifies relationship between each word and generate incorrect prediction accordingly. This problem might be the attribute of QANet architecture which rely on CNN for local dependency that have limited search neighborhood compare to RNN that scan entire context. Therefore, additional feature from dependency or long-term dependency (e.g. Transformer-XL [Dai et al., 2019]) could prove to be useful to resolve this issue.

6 Conclusion

In this paper, we introduce QANet with Universal Transformer that build on top of QANet architecture. We incorporate recurrent transformation through time by using time encoding and weight sharing of convolution and feed-forward layer in encoder block for both embedding encoder layer and model encoder layer. This architecture directly tackles the shortcoming of Transformer-based encoder which is an inductive bias from recurrent structure that are useful to natural language understanding task and improve the model performance substantially. The error analysis reveal that most of incorrect prediction is attribute to dependency structure of context and query. Further improvement on this architecture is to incorporate Adaptive Universal Transformer [Dehghani et al., 2018] that dynamically determined suitable recursion in encoder block instead of fixed recursion depth in Universal Transformer.

References

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

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019. URL `http://arxiv.org/abs/1901.02860`.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *CoRR*, abs/1807.03819, 2018. URL `http://arxiv.org/abs/1807.03819`.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. *CoRR*, abs/1603.09382, 2016. URL `http://arxiv.org/abs/1603.09382`.

Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL `http://arxiv.org/abs/1408.5882`.

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

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL `http://arxiv.org/abs/1606.05250`.

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`.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015. URL `http://arxiv.org/abs/1505.00387`.

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, 2017. URL `http://arxiv.org/abs/1706.03762`.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018. URL `http://arxiv.org/abs/1804.09541`.