Self-attention and convolution for question answering on SQuAD 2.0: revisiting QANet

Stanford CS224N Default Project

Oualid El Hajouji
ICME
Stanford University
oualidel@stanford.edu

Joachim Sasson
ICME
Stanford University
jsasson@stanford.edu

Abstract

In recent years, significant work in NLP research has been done to build Question Answering systems. QANet was the first QA model that combined self-attention and convolution, without any use of Recurrent Neural Networks. The goal of this project is to tackle the Question Answering task on the SQuAD 2.0 dataset using different variations of the QANet architecture. We first re-implement the QANet model, and then explore different versions of the architecture. We propose several ensemble models with different inference methods: our best model, using a two-step answerability prediction based inference method, achieves 71.21 F1/ 68.14 EM on the development set, and 69.04 F1 / 65.87 EM on the test set.

1 Introduction

The problem of machine reading comprehension has sparked interest in the NLP community, partly thanks to the availability of public datasets such as SQuAD [1]. One of the main tasks in MRC is Question Answering, defined as follows: given a question and a paragraph as inputs, find a subset of the paragraph that answers the question. In this work, we consider the more general formulation, in which a question can have no answer in the paragraph. One great interest of this problem is that it quantifies how well deep learning systems can 'understand' text all the while enabling an automatic understanding of pieces of text, which can be extremely useful in many industries.

The two main approaches developed to tackle Question Answering are Pre-trained Contextual Embeddings (PCE) based methods, such as Bert [2] and ELMo [3], and Non-PCE methods [4, 5]. On the SQuAD 2.0 dataset, the state of the art methods are PCE based, even though some non-PCE methods are still successful in solving the task. One of those methods, QANet [5], is the main focus of this work.

QANet, at the time of publication, achieved state of the art results on the SQuAD 1.1 dataset all the while significantly improving efficiency thanks to the use of a convolution and attention-based [6] architecture (as opposed to Recurrent Neural Network based architectures). Here, we choose to adapt the QANet method to the SQuAD 2.0 dataset, where questions can have no answer, and explore different variations of the QANet architecture, especially regarding the size of the model, the attention mechanism and the parametrization of the model. We also explore different ways of ensembling our single models, with a focus on different inference methods. To the best of our knowledge, our best performing ensemble inference method, which relies on an answerability prediction step, has not been proposed in previous work.

2 Related work

Before QANet [5], most methods used for Question Answering relied on Recurrent Neural Networks, along with a use of attention mechanisms. Among these methods, there is the Gated Self-Matching

Networks [7] method and the Bidirectional Attention Flow (BiDAF) [4] method. The latter is a hierarchical multi-stage architecture that allows modeling of the context paragraph at different levels of granularity: character-level word embeddings and word-level embeddings. It employs recurrent units such as LSTM to keep track of sequential information in the text as well as attention mechanism in order to get long term dependencies. More precisely, the attention process is divided into two parts: Context-to-Question (context aware attention) and Question-to-Context (question aware attention). BiDAF was among the best performing models for the QA task when it was first published. However, since it is RNN based, it has the inconvenient of being computationally costly.

The necessity to adopt a radical change of structure was concretized in 2018 by the authors of the QANet model [5]. This model, similarly to the Transformers architecture [6], does not use LSTM units at all. It encodes the question and context separately thanks to a non-recurrent Embedding Encoder Layer which uses convolution to get local information on the input, and self-attention to get long term interactions. The non-recurrent character of this network makes it much faster than the BiDAF model, with a 3x to 13x improvement in training speed as reported by the authors.

While QANet has achieved state of the art performance, there are still variations of the architecture that were not explored in the original paper. First of all, it was not originally implemented for the SQuAD 2.0 framework, so the original version cannot predict answerability. Moreover, a variation that could potentially improve performance is in the self-attention mechanism: the original paper does not experiment with any other attention method than dot-product attention. Multiplicative attention, which we explore in this work, could enable to better capture global interactions. Also, the QANet authors chose to share the parameters between the components of their Model Encoder layer: here, we explore the case when those parameters are unshared.

3 Approach

As outlined in previous sections, we tackle the Question Answering task by implementing from scratch the QANet model described in [5]. Our implementation was made easier thanks to the provided BiDAF implementation, since it has some similarities with the QANet, and because it is already adapted for SQuAD 2.0. Then, we explore variations of the original QANet, described in this section. Finally, we propose different ensemble methods of our single models.

3.1 Baseline

We used the Bidirectional Attention Flow (BiDAF) [4] as a baseline model. The version does not use character embeddings.

3.2 QANet architecture

Let us provide a description of the original QANet algorithm by following the progression of figure 1. First, contexts and questions are processed separately: their words go through an **Input Embedding Layer**, which produces a concatenation of a fixed GloVe word embedding and character-based word representation. On top of that, a two-layer highway network (as described in the handout) refines the embedding and produces the hidden states q_i and c_i.

This phase is followed by an **Embedding Encoder Layer** which takes the embeddings q_i and c_i and the sinusoidal position encodings as inputs, and stacks blocks each composed of 4 depthwise separable convolutional layers, a self-attention layer (multi-head) and a feed-forward layer. This phase is crucial to the understanding of the algorithm. By analogy with their use in computer vision, the role of the convolutional layers is to get local information in the inputs. The fundamental idea behind depthwise separable convolutions is to separately operate a spatial feature learning step (depthwise convolution) and a channel combination step (pointwise convolution). More details about this method are provided in [8]. As for the multi-head self-attention layer described in [6], it calls the query for each position in the input and computes a weighted sum of all positions, or keys, in the input based on the similarity between the query and key.

Then, there is a **Context-Query Attention Layer**, which computes pairwise similarities between context and query words and then computes the context-to-query and query-to-context attention distributions and the attention output. More precisely, the computations made by this layer are:

- *Context-to-query attention*: Denoting C the encoded context and Q the encoded query, we compute S a matrix of pairwise similarities. These similarities are computed using a trilinear function [9] $f(q, c) = W_0[q, c, q \odot c]$. The context-to-query attention is then $A = \text{softmax}_{row}(S)\ Q^T$, softmax_{row} referring to softmax row normalization.
- *Query-to-context attention*: The query-to-context is then computed as $B = S\ \text{softmax}_{col}(S)\ C^T$ (softmax_{col} being sofrmax column normalization)
- *Attention output*: The output of the layer is $[C, A, C \odot A, C \odot B]$

It is followed by a **Model Encoder Layer**, with 3 blocks similar to those of the Embedding Encoder Layer that share their weights.

Finally, the **Output Layer** computes the probability of each position in the context being the start ($p_{start}(i)$) or the end ($p_{end}(i)$) of the answer span using the first two blocks of the model encoder layer for p_{start} and the first and third block for p_{end}. That is to say, denoting M_0, M_1, M_2 the outputs of the three blocks of the Model Encoding Layer, we have:

$$p_{start} = \text{softmax}(W_1[M_0, M_1]) \qquad p_{end} = \text{softmax}(W_2[M_0, M_2])$$

The loss for an example of which the answer span starts at i and ends at j is:

$$-\log p_{start}(i) - \log p_{end}(j)$$

At inference time, similarly to the BiDAF model, the predicted start and end indexes are those that maximize the joint probability $p_{start}(i)p_{end}(j)$, given that the indexes i and j are such that $0 \leq j - i \leq M$ with M a chosen parameter representing the maximum length of an answer. The case $j = i$ is possible only when both indexes are 0, in which case the model predicts that there is no answer (each context's first word is an out-of-vocabulary token used for the purpose of predicting if there is an answer).

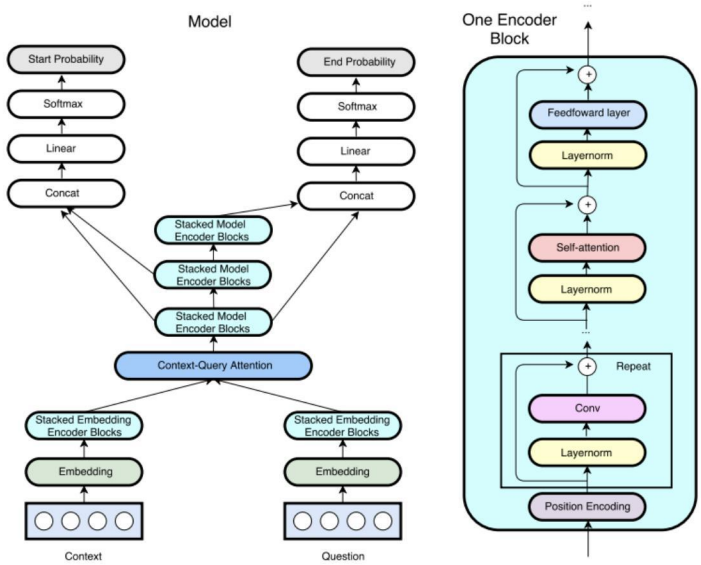

Figure 1: QANet architecture (figure from [5])

3.3 Architecture variations

We experimented with several variations of the QANet architecture.

- **Model size**: Due to computational constraints at the beginning of the project, the first QANet model we implemented was smaller than the original one, as it had less blocks in the model encoder layer and less heads for the multi-head self-attention in the model and embedding encoder layers. We refer to this model as *Small QANet*. The second QANet model we implemented, *Big QANet*, is identical to the original one. More details are given in subsection 4.2.
- **Attention mechanism**: The original QANet uses dot-product attention only for the self-attention mechanism in the encoder blocks of the model and embedding encoder layer. One of the models we implemented, *Multiplicative QANet*, replaces dot-product attention with multiplicative attention. The intuition behind this modification was that multiplicative attention would be able to capture more complex global interactions than dot-product attention, since it allows the model to learn the interaction terms.
- **Unshared parameters** : The original QANet shares the parameters between the 3 blocks of the model encoder layer M_0, M_1, M_2. We experimented with an architecture where those parameters are unshared, thus having 3 different sets of parameters for the 3 blocks. It was unclear to us why sharing the parameters would be beneficial for the model to learn a proper probability distribution of the answer span, and we therefore chose to trade potential bias for variance.

3.4 Ensemble models

We implemented 3 ensemble models which used the single models we trained (including the baseline), based on 3 different inference methods. All 3 only use the predicted answer spans of the single models: they do not rely on the probability distributions, as we can sometimes see in the literature.

- **Majority voting**: This ensemble model predicts an answer by taking the most common predicted answer among the answers given by the single models. In the case of a tie, we choose the answer among the most common ones outputted by the best performing single model (in terms of F1 score).
- **Answerability prediction (1)**: We propose a two-step inference method for this ensemble model. We first select the single model, denoted f^{*1}, with the lowest False Negative rate in the answerability prediction (negative corresponding to no answer). For a given example (c, q), if $f^{*1}(c, q) =$ `<NOANSWER>`, the ensemble model outputs no answer. Else, the model gives the output obtained with majority voting. The rationale behind this inference method is that, when f^{*1} outputs no answer, the probability that the question is not answerable is relatively high, so the ensemble model can output no answer with high confidence.
- **Answerability prediction (2)**: This ensemble model is very similar to *Answerability prediction (1)*, except that the selected reference model for answerability prediction is the one that has the highest accuracy (AvNA) in the answerability prediction task.

4 Experiments

4.1 Data and evaluation

We use the SQuAD 2.0 dataset and some hand-labeled examples (for the test set) with the following train/dev/test split:

1. `train`: 129941 examples from the official SQuAD 2.0 training set
2. `dev`: 6078 examples from the official dev set
3. `test`: 5915 examples either from the official dev set or hand-labeled

Figure 2 shows the distribution of the first words of the questions in the development set for the 10 most frequent first words ("Other" representing all other first words). The first word is a proxy for the question type. We can see that there is a majority of "What" questions.

The numerical evaluation metrics we use are the Exact Match (EM) and the F1 score. Also, the AvNA (Answer vs No Answer) refers to the accuracy of the prediction of answerability.

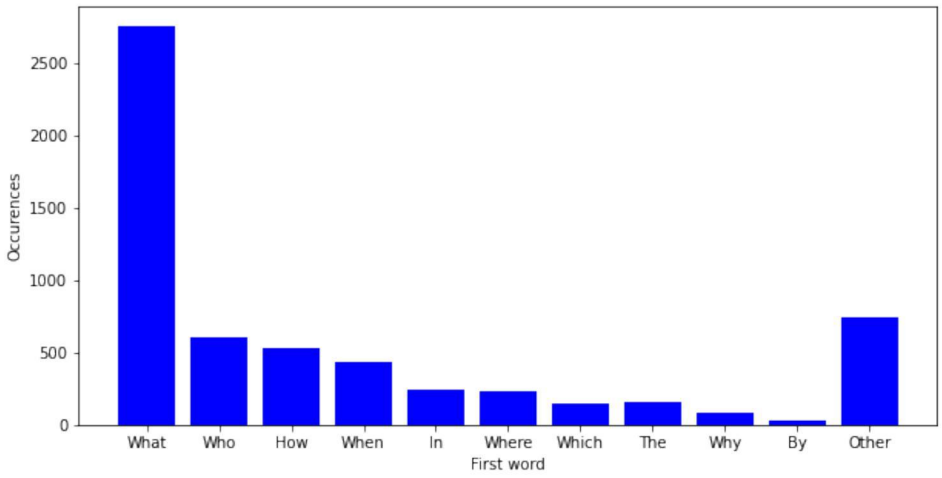

Figure 2: Distribution of questions according to their first word

4.2 Experimental details

First, here are the parameters that were unchanged for all our models, and that correspond to the ones in [5]:

- The numbers of convolution layers in the Embedding and Modeling encoder are 4 and 2, kernel sizes are 7 and 5.
- `hidden_size` $= 128$, `n_epochs` $= 30$.
- We used an ADAM optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$.
- We applied an exponential moving average on all trainable parameters with a decay rate 0.9999.
- We used a learning rate warm-up scheme with an inverse exponential increase from 0 to 0.001 in the first 1000 steps, and then constant learning rate.
- We used L2 weight decay with parameter $\lambda = 3 \times 10^{-7}$
- Dropout rate between layers is 0.1, word dropout rate is 0.1 and character dropout rate is 0.05.
- We used the stochastic depth method [10]: sublayers l of each embedding or model encoder layer are dropped out with a rate $0.1\frac{l}{L}$ (L is the final layer).

For *Small QANet* and *Multiplicative QANet*, we used 4 heads for self-attention, and 3 encoder blocks in the Model Encoder layers. The batch size used was 32. Those models trained in approximately 20 hours.

For *Big QANet* and *Unshared QANet*, we used 8 heads for self-attention, and 7 encoder blocks in the Model Encoder layers, similarly to what is done in [5]. The batch size used was 16. Those models trained in approximately 40 hours.

All ensemble models used our 4 single models, along with a "copy" of *Small QANet* and *Big QANet*, defining copy as the same model independently trained. That is to say, we trained *Small QANet* and *Big QANet* twice, thus obtaining 2 different sets of weights for each of these two models.

4.3 Results

For each of our single models and ensemble models, we report in Table 4.3 the F1 score, EM, AvNA, number of False Positives (FP), True Positives (TP), False Negatives (FN) and True Negatives (TN) on the development set. The best metrics, for single models and ensemble models, are in bold.

Our best model, the ensemble model *Answerability prediction (2)*, scored **69.04 F1/65.87 EM** on the **IID SQuAD track** test set, which gives it the 4^{th} rank on the test leaderboard. The differences in dev

Model	F1	Exact Match	AvNA	FP	TP	FN	TN
BiDAF	59.44	56.02	66.12	1492	2324	524	1611
Small QANet	66.86	63.91	73.38	**995**	2259	589	**2108**
Multiplicative QANet	66.36	62.51	73.20	1160	**2413**	**435**	1943
Big QANet	68.66	**64.93**	74.73	1010	2354	494	2093
Unshared QANet	**68.73**	64.88	**75.03**	1023	2385	463	2080
Majority voting	70.50	67.33	75.82	939	**2348**	**500**	2164
Answerability prediction (1)	70.79	67.74	75.89	839	2252	596	2264
Answerability prediction (2)	**71.21**	**68.14**	**76.29**	**802**	2239	609	**2301**

Table 1: Comparison of models' performances on the development set

and test scores are probably due to differences in data distribution, as well as model selection which was done on the dev set.

First, we observe that all our models largely outperformed the baseline BiDAF model, which is consistent with what we observe in the literature. However, the baseline could largely be improved, for example through the use of character embeddings. While we cannot conclude with certainty that the QANet-based models would outperform the improved BiDAF models, results already observed indicate that they probably would. This performance gap was expected: the QANet models are probably better at having both local focus and global understanding of the text.

Second, we notice that the bigger versions of the model outperform the smaller ones with a +1 margin on the EM and a +2 margin on the F1 score. Having significantly more parameters, with an increased number of heads and number of blocks in the model encoder layer, helped the bigger models learn a better probability distribution of answer spans. While those bigger models are more prone to overfitting, this does not seem to have affected negatively the performance on the development set.

Our variations of the architecture of QANet did not clearly outperform the original versions. Due to the model size of our models, the effects of these variations can only be observed through the comparison of *Small QANet* and *Multiplicative QANet* on one side, and *Big QANet* and *Unshared QANet* on the other. *Multiplicative QANet* was outperformed by *Small QANet*, which is surprising since we expected the multiplicative self-attention to enable a better modeling of the interactions. This might be due to overfitting, since multiplicative self-attention adds a lot of parameters to the model, and due to the fact that hyperparameters (such as dropout parameters or l2 regularization factor) were chosen for the original QANet in [5]. Due to time and computational constraints, we could not tune the hyperparameters for all our models. However, we can observe that *Multiplicative QANet* has the best False Negative and True Positive rate: it seems to predict that there is an answer more often than other models, but relatively reasonably since its FP and TP are significantly better than those of BiDAF.

As for the *Unshared QANet*, it has a slightly higher F1 score and AvNA than *Big QANet*, but a slightly lower EM. The performances are therefore quite similar, and it is difficult to know whether the small differences observed are random or if they are due to real differences in predictive performance. Again, this model is more prone to overfitting and hyperparameter tuning could help improve its performance.

As expected, the ensemble models perform significantly better than single models. The best ensemble model, *Answerability prediction (2)*, has a +0.7 F1 score margin and +0.8 EM margin on *Majority Voting*. The idea of separating answerability prediction and answer prediction using the best classifier seems to be the best one for inference: it is a convenient way of making the model easily classify examples which have no answer.

5 Analysis

In this section, we perform the analysis of our best performing model.

Figures 3 and 4 show the metrics obtained by model on different subsets of the data, those subsets being defined by the first words of the questions. First, we can observe that the easiest examples for our model are the ones with questions that start with "When". This is expected since those questions have numerical answers most of the time, which can be more easily spotted by the model (errors

Figure 3: AvNA with respect to questions' first word

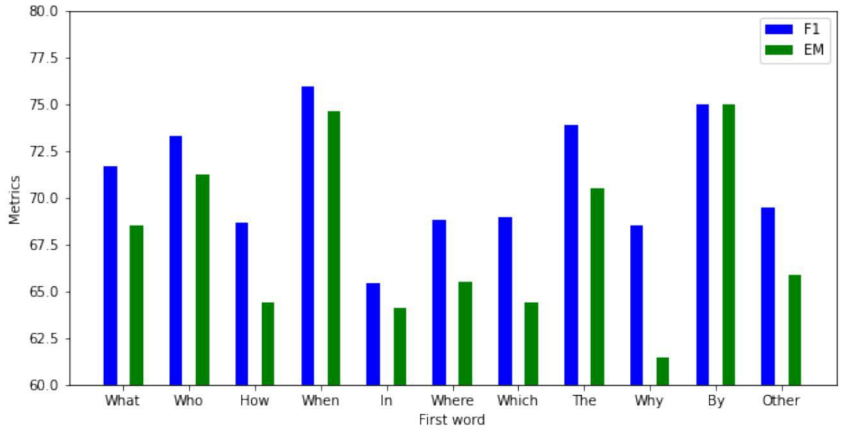

Figure 4: F1 score and EM with respect to questions' first word

probably happen when there are several numerical substrings in the context). The model also has good performance on "Who" questions, probably for similar reasons than for "When" questions: the answers to these questions are often a name.

The model makes more errors for "How", "Where" and "Why" questions, which is not surprising for "How" and "Why" questions since they require logical reasoning, which is a hard skill for an NLP model to learn. Interestingly, the gap between F1 score and EM for "Why" questions is very big: the model seems to be able to output an answer that overlaps with the ground truth, meaning that it has a reasonable knowledge of where the answer approximately is, but it has trouble identifying it exactly. For "How" questions, we can see that the AvNA is among the lowest: the model has trouble detecting if there is an answer, as if it did not really understand the question.

In Table 5, the first example illustrates our last claim: this example, a "How" question, requires logical reasoning, which makes it hard for the model to make a prediction, as it rather outputs no prediction. The second example shows that the model does understand that a "How many" question calls for a numerical answer, but it selected the only one in the text and it was not correct. The model's understanding of the context-query was limited to the format rather than the meaning. The third example shows that the model made a semantic error, probably due to erroneous context-query matching based on the similarity of the query word "bacteria" and "bacteriophages".

Context	Question	Prediction
Forces act in a particular direction and have sizes dependent upon how strong the push or pull is. Because of these characteristics, forces are classified as "vector quantities". For example, when determining what happens when two forces act on the same object, it is necessary to know both the magnitude and the direction of both forces to calculate the result. For example, if you know that two people are pulling on the same rope with known magnitudes of force but you do not know which direction either person is pulling, it is impossible to determine what the acceleration of the rope will be. **Associating forces with vectors** avoids such problems.	How do you determine the acceleration of a rope when two people are pulling it?	
As interesting examples of expositions the most notable are: the world's first Museum of Posters boasting one of the largest collections of art posters in the world, Museum of Hunting and Riding and the Railway Museum. From among Warsaw's 60 museums, the most prestigious ones are National Museum with a collection of works whose origin ranges in time from antiquity till the present epoch.	How many posters are in Warsaw?	60
Immune systems appear even in the structurally most simple forms of life, with bacteria using a unique defense mechanism, called **the restriction modification system** to protect themselves from viral pathogens, called bacteriophages. Prokaryotes also possess acquired immunity, through a system that uses CRISPR sequences to retain fragments of the genomes of phage that they have come into contact with in the past, which allows them to block virus replication through a form of RNA interference.	What is the main defense mechanism of bacteria known as?	bacteriophages

Table 2: Error examples

6 Conclusion

In this work, we designed and evaluated several models for the Question Answering task on the SQuAD 2.0 dataset, all based on the QANet architecture. We re-implemented the QANet from scratch and experimented with different variations of its architecture, and then designed several ensemble methods with different inference methods.

In terms of results, our models significantly outperformed the RNN-based baseline BiDAF, and our variations did not yield significant performance differences compared to the original QANet method, even though we did observe some interesting differences. A hyperparameter tuning phase might be necessary to have a better understanding of whether those changes to the original model can help it perform better. Our ensemble models achieved very good performance, especially the *Answerability prediction (2)* model, which was based on a novel inference method. Future work might include hyperparameter tuning as well as other architecture modifications. Also, it might be key to train a better version of BiDAF to build a stronger ensemble model.

References

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

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

[3] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[4] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[5] 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. *arXiv preprint arXiv:1804.09541*, 2018.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[7] 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 (Volume 1: Long Papers)*, pages 189–198, 2017.

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

[9] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making neural qa as simple as possible but not simpler. *arXiv preprint arXiv:1703.04816*, 2017.

[10] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.