Investigating QANet's Convolution Layer

Stanford CS224N Default Project (IID SQuAD Track)

Andy Jin
Department of Computer Science
Stanford University
andyjin@stanford.edu

Matthew Early
Department of Computer Science
Department of Linguistics
Stanford University
mmearly@stanford.edu

Jesse Doan
Department of Computer Science
Stanford University
jdoan21@stanford.edu

Abstract

Tasked with building a model for Question-Answering on the SQuAD 2.0 dataset, we re-implement QANet and perform several extensions to boost performance. The motivation of QANet is that it replaces the RNN in traditional QA models (like BiDAF) with convolutions, which can lead to faster training and inference. Our further experiments investigate the purpose of QANet's convolution layer by assessing performance when swapping it with local and global self-attention layers. This contributes to an understanding of whether the convolution layer does indeed capture local structure in sentences, as the original authors of QANet posited. We observed small improvements by changing the number of blocks in each layer of model encoders, and our best performance was achieved when creating an ensemble of results from QANet models of various model encoder layer block sizes (+8.4 F1 and +8.6 EM compared to BiDAF baseline). The experimental model that used repeated local self-attention layers instead of convolution layers outperformed the model that replaced the convolution layers with global self-attention layers, supporting the original authors' hypothesis that the convolution layer functions primarily to capture local structure.

1 Key Information to include

Mentor: **Ben Newman**. External Collaborators (if you have any): **None**. Sharing project: **None**.

2 Introduction

In 2018, a paper titled "QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension" showcased results on the SQuAD dataset with faster training and faster inference while achieving equivalent accuracy as the state-of-the-art at the time [1]. As a quick refresher, the task of reading comprehension or question answering of the SQuAD dataset is to take in a paragraph as an input along with a question about that paragraph. Then, the model outputs the correct start and end positions in the context that highlight the answer. We can see an example of this in Figure 1. QANet builds on the Bidirectional Attention Flow (BiDAF) model [2], which we use as our baseline model and uses a recurrent model and an attention component.

Due to the slow sequential nature of RNNs for both training and inference, the authors of QANet proposed a new architecture to address this challenge. They did not use recurrent networks but rather an encoder that consists of convolution and self-attention, leading to a speedup in performance. The

Question: Why was Tesla returned to Gospic?
Context paragraph: On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. On 17 April 1879, Milutin Tesla died at the age of 60 after contracting an unspecified illness (although some sources say that he died of a stroke). During that year, Tesla taught a large class of students in his old school, Higher Real Gymnasium, in Gospic.
Answer: not having a residence permit

Figure 1: Example of SQuAD Question, Context, & Answer

authors hypothesize that convolution captures the local structure of the text while the self-attention learns the global interaction between each pair of words.

In our paper, we implement QANet per the paper specifications where we achieve an approximate increase of 6.3 for EM and 6.5 for F1 compared to our baseline BiDAF model. Further, we extended our model to utilize character embeddings and experimented with a different number of blocks in the model encoder layer. We also investigated QANet's convolution layer by experimenting with the removal of convolutions and substitution of additional attention. Finally, we constructed an ensemble model which outperformed all of our other models (+8.6 EM and +8.4 F1 compared to BiDAF).

3 Related Work

Before QANet, work to solve the Question Answering task was dominated by recurrent models that modeled the complex interactions between the context paragraph and question. One example of this is the Bi-Directional Attention Flow (BiDAF) model (Seo et al.), published in 2016 which introduced the idea of bi-directional flow of attention from context to query and from query to context while achieving state-of-the-art performance for its time [2]. Typically before BiDAF, methods that used attention focused only on the attention flowing from the question to the context.

After BiDAF, the introduction of the Transformer showcased state-of-the-art results on the WMT 2014 English-to-German translation task by using a novel architecture based solely on attention mechanisms (Vaswani et al.) [3]. With the removal of recurrence and convolutions entirely (which were the dominant models at the time), the Transformer not only surpassed in performance but also was more parallelizable and took significantly less time to train compared to prior models.

Building on the work of Transformers, QANet made a significant impact as it eliminated the need of recurrent models specifically in question-answering on the SQuAD dataset, achieving state-of-the-art results. Our work is a nice next step as we replicate the relative success of QANet over RNN-based models, and further distill the precise role of the QANet convolution layer and effects of ensembling variations of QANet.

Later in 2018, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al.) demonstrated that non-RNN-based models can achieve even higher efficiency and accuracy [4]. It is especially promising for the future of NLP since we have the ability to train on significantly more data within the same time window it takes to train an LSTM, but still maintain high accuracy.

4 Approach

Baseline: Our baseline was the provided BiDAF baseline model, which used only word embeddings. BiDAF is described in [2].

Approach #1: Add Character Embedding to Input Embedding Layer

We loaded the provided character embeddings and added them to our BiDAF baseline by concatenating them to the word embeddings. However, this led to poor results, so we enhanced the character embeddings by feeding them through a convolution layer before max pooling. Then, the concatenated tensor of the word and character embeddings were sent through a 1D convolution layer (since the tensor is 3D, not 4D), and finally the highway network. We utilized character embeddings in all of our QANet models as well. We tried several variations of the convolutions (along with different kernel sizes and paddings) delineated in Table 1:

Model	# Epochs	CharEmb	Kernel Size	Proj before Highway	Kernel Size
Baseline BiDAF	30	N/A	N/A	Linear	N/A
BiDAF w/ CE 2d, $k = 5$	17	Conv2d	(1, 5)	Conv1d	1
BiDAF w/ CE 1d, $k = 1$	30	Conv1d	1	Conv1d	1
BiDAF w/ CE 2d, $k = 1$	13	Conv2d	1	Conv1d	1
QANet w/ CE 2d, $k = 1$	30	Conv2d	1	Conv1d	1

Table 1: List of Character Embedding (CE) Enhanced Models

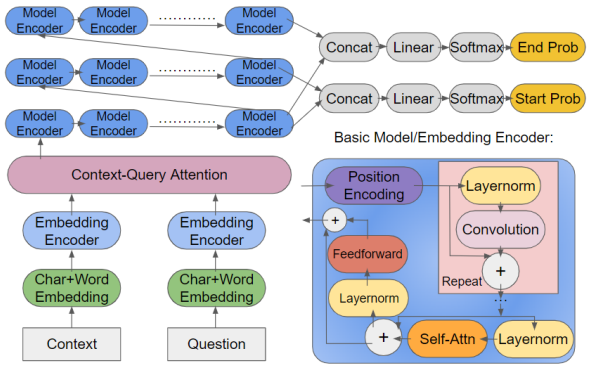

Figure 2: QANet Model Architecture

Approach #2: QANet Implementation with Character Embedding

1. **Positional Encoder:** As described by the QANet paper [1], we implemented a positional encoder from scratch, consisting of *sin* and *cos* functions at various wavelengths defined in [3]. Namely, we added the input to the positional encoder at the start of each encoder layer.
2. **Replace RNN Encoder with QANet Encoder Blocks:** Again closely following the QANet paper, we implemented the stacked embedding encoder and model encoder blocks, which consisted of the positional encoder, layernorm, self-attention (for which we tried the causal self-attention from Assignment #5 as well as the Pytorch multihead attention module), and a feedforward layer. In this step, we also concatenated the first and second, and first and third stacked model encoder block outputs, and implemented the output layer (linear + softmax) to generate our start and end probabilities, respectively. Note that we used the same context-query attention as the baseline BiDAF.
3. **Convolution layers in Encoder block:** For the implementation presented in [1], the bulk of the "brain power" in each encoder block comes from a series of convolution+layernorm layers that precede the self-attention and feed-forward layers built in our previous step. We took advantage of `nn.ModuleList` to construct a series of Depthwise Separable Convolution layers followed by layernorm layers, using the parameters specified by [1] (128 filters, kernel size 7, 4 convolution layers for blocks in the embedding encoder layer and 2 layers for blocks in the model encoder layer). Per the QANet paper, depthwise separable convolutions are memory efficient and have better generalization than traditional convolutions. In this step, we also added the residual block (e.g., $f(layernorm(x)) + x$ for operation f), which preserves an identity path from the input to output of each encoder block.
4. **Dropout—Stochastic Depth Method:** We implemented the stochastic depth method of dropout for each layer in the embedding and model encoder [5]. In particular, we keep each sublayer l in the encoder with probability $1 - \frac{1}{L}(1 - p_L)$ where L is the last layer and $p_L = 0.9$. We also added dropout (with $p = 0.1$) between every two layers.
5. **Output Layer:** In making our predictions, we choose a start and end position in the context paragraph with probabilities p^1, p^2, respectively: $p^1 = softmax(W_1[M_0; M_1])$ and $p^2 = softmax(W_2[M_0; M_2])$, where M_0, M_1, and M_2 are the outputs from the three model encoders in order from first to last and W_1 and W_2 are the trainable variables. The

score of a prediction (e.g. a start and an end position) is the product of the probabilities of the start position and of the end position. Our objective function is the negative sum of the log probabilities of our prediction distributions using the correct start and end position, averaged over all training examples: $L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\log(p^1_{y_i^1}) + \log(p^2_{y_i^2})\right]$, where y_i^1 and y_i^2 are the ground truth indices for the start position and the end position for example i with trainable variables captured by θ.

We can see the above elements of the QANet model architecture in Figure 2.

Approach #3: Modifying Model Encoder Stack Size

We experimented with upscaled and downscaled versions of QANet with 4, 5, 6, and 9 model encoder blocks (in lieu of the original 7). The goal is to explore if altering the encoder stack size would significantly impact performance.

Approach #4: QANet without convolutions

Instead of convolution layers in the encoder block, we used a *novel approach* that replaced them with local self-attention and "global" self-attention. By measuring performance of these models, we can assess the authors' hypothesis that convolution layers capture local interaction in text. Under this hypothesis, we expect better performance on a model that substitutes convolution layers for extra local self-attention than a model that substitutes convolution layers for extra global self-attention, as the former is more structurally in line with the assumed purpose of the convolution layer.

Approach #5: QANet ensemble

Lastly, we used the predictions of seven QANet models to create an ensemble prediction for a given context-question pair. We applied a majority vote from the following 7 models: QANet with 4 blocks, QANet with 5 blocks (2x, run with different random seeds), QANet with 6 blocks, QANet with 7 blocks (2x, run with different random seeds), QANet with `dropout` $= 0.15$. Ties were broken according to the model that highest dev F1. Our reasoning with the ensemble model (which combined our best performing models) is that it can reduce variance and increase accuracy by taking into account multiple well-performing models instead of relying on one model.

5 Experiments

5.1 Data

We used the SQuAD 2.0 dataset provided to us, which contains answerable and non-answerable questions. There are 129,941 training examples (from the official train set), 6,078 dev examples (about half of the official dev set), and 5,915 test examples (about the other half of the official dev set). We also used the provided pre-trained 300-dimensional GloVe vectors as model inputs.

5.2 Evaluation method

We used the evaluation method as described in the SQuAD handout: F1 score and Exact Match score (EM). During training, we also monitor the Answer vs. No Answer classification accuracy (AvNA) and the dev set's negative log-likelihood (NLL), but these are not the core metrics for evaluation.

5.3 Experimental details

All of our experiments were run Microsoft Azure GPU using Pytorch. Our hyperparameters, inspired by the QANet paper, include: **L2 weight decay** ($\lambda = 3 \times 10^{-7}$) on all trainable variables. **Hidden size of 128**: We also tried increasing hidden size by 50% (to 192), but this did not meaningfully improve results. **Convolution filter number** of 128. **Batch size** of 16: We tried a batch size of 32, but this did not meaningfully improve results. **ADAM optimizer** (with $\beta_1 = 0.8, \beta_2 = 0.999, \epsilon = 10^{-7}$). **Learning rate of 0.001**, using a warm-up framework with inverse exponential increase from 0 to

Model	F1 (Dev)	EM (Dev)
Baseline BiDAF	58	55
BiDAF w/ Char Embed 2d, $k = 5$	62.81	59.62
BiDAF w/ Char Embed 1d, $k = 1$	63.25	59.82
BiDAF w/ Char Embed 2d, $k = 1$	63.57	60.39
Basic QANet (`n_blocks`=7)	64.49	61.27
QANet, ensemble	**66.42**	**63.57**
QANet, `n_blocks`=4	63.86	60.59
QANet, `n_blocks`=5	65.67	62.34
QANet, `n_blocks`=6	63.96	60.85
QANet, `n_blocks`=5, `dropout`=0.15	61.09	58.07
QANet, stochastic dropout `survive_prob`=1	52.19	52.19
QANet, replace conv with local self-attention	52.19	52.19
QANet, replace conv with global self-attention	51.89	51.87
QANet, no convolutions	50.84	50.55

Table 2: Dev EM and F1 Scores

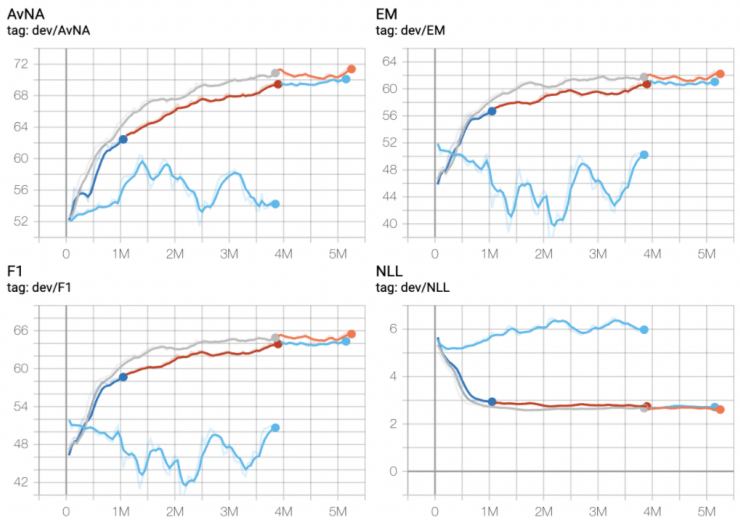

Figure 3: Best Model Training Curves

0.001 in the first 1,000 steps. **Exponential moving average** (decay rate of 0.999) applied on all trainable variables. **Dropout** rate of 0.1 on word embeddings, 0.05 on character embeddings, 0.1 between every two layers, and the stochastic depth method within each embedding encoder and model encoder layer.

Other model configurations, such as changing the number of model encoder blocks (3, 4, 5, 7, and 9) were described in Section 4. Regarding training time, the improvements on the baseline BiDAF model (e.g., changing the type of convolution and kernel size) each took about 5 to 8 hours to complete 30 epochs. QANet models took about 15 hours for 30 epochs and about 20 hours for 40 epochs. This is slower than expected, given that the original QANet paper motivated that QANet may lead to faster performance than RNN models since it can be easily parallelized. We hypothesize that our slower results are due to the limited memory we have on Azure.

5.4 Results

Table 2 shows the F1 and EM results for each experiment on the dev set. We submitted our two best models, QANet with 5 blocks and the ensemble QANet to the test leaderboard for the IID SQuAD track. **QANet with ensemble** had F1 score of 63.82 and EM score of 61.10. **QANet with 5 blocks** had F1 score of 62.84 and EM score of 57.76.

In Figure 3 training curves, the top-most (grey and orange) line on the AvNA plot depicts the downsized QANet with 5 encoder blocks. The middle line on the AvNA plot (blue, red, and cyan) depicts the basic QANet with 7 encoder blocks. The bottom line on the AvNA plot (cyan) depicts the QANet where convolution is replaced with a global attention and feedforward layer.

When character embeddings are added, our models are able to outperform the BiDAF baseline. This is expected because character embeddings allow us to capture the internal structure of words and better handle out-of-vocabulary words, which allows the model to more accurately infer the start and end positions of the answer. It is also encouraging (and expected) that our basic QANet outperforms the baseline BiDAF, although the results are a bit lower than the expected EM and F1 scores of 73.6 and 82.7 (respectively) which were presented in the original QANet paper. We suspect this may be due to some implementation differences not directly revealed in the paper, as we made sure to match the configurations delineated in the paper. That said, our results are comparable to others on the leaderboard, and are confident that there is not a coding error.

Among our QANet extension models, the ensemble model performed the best on both the dev and test set, which makes sense as its predictions are less sensitive to aberrations of individual models. By taking a majority vote among seven models, we reduce the variance of the predictions.

On the other hand, we found that upscaling and downscaling QANet did not significantly improve performance. Though QANet with 5 encoder blocks had the highest EM and F1 scores, it was not by a large margin, and they were similar to scores if we used 4, 6, 7, or 9 blocks. These results suggest that changing the complexity of QANet does not significantly help its performance; instead, more effort should be placed on enhancing the internal model architecture.

We investigated the convolution layer of the QANet and sought to replace it with a layer of global self-attention. The dev NLL began to increase very early in training, so we discontinued this experiment and added a feedforward layer after the self-attention, mimicking the architecture in the original encoder block. Re-running the experiment, the NLL was relatively more reasonable, although it still increased over time. With a local layer of self-attention in lieu of convolutions, the EM and F1 scores were higher than those for global self-attention. However, both of these substitutions had lower EM and F1 than the QANet with convolutions, which is a sign that convolution is indispensible to the QANet architecture in capturing local structure within sentences.

Other modifications such as changing the convolution to be a 1D conv versus 2D conv, changing the kernel size, and changing the dropout rate, did not significantly improve performance. Thus, we propose the next step to be fundamentally changing the training scheme and QANet architecture, such as by using data augmentation via back-translation to create a richer training set, or by modifying the output layer to condition the end probability on the start probability.

6 Analysis

We are very interested by the results that come from changing the number of model encoder blocks within each of the three stacks. Despite the original paper's choice to use 7 blocks per stack, our best (non-ensemble) model was one that used 5 blocks in each stack. While this may initially suggest that fewer blocks result in better performance, our further analysis reveals that though performance improves when going from 7 blocks to 5, performance decreases when going from 7 blocks to 6 and further decreases when going from 7 blocks to 4. Attempts to use 9 blocks also did not improve performance, implying that there is a "sweet spot" at around 5 encoder blocks per stack.

Our results shed light on the original paper's hypothesis for the function of the convolution layer. First, to confirm that the convolutions themselves indeed contributed to the overall model performance, we compared our implementation of QANet as described in the paper to a version that had the convolutions removed entirely (See Table 2: "QANet, no convolutions"). This caused a significant drop in F1 & EM ratings (-13.65 and -10.72), demonstrating that the additional "brain power" provided by the layer of convolutions is crucial for good performance (and more significant than the small improvements observed from tweaking the number of encoder blocks per layer).

QANet, ensemble		
Pred/Truth	Answer	No Answer
Answer	2122	992
No Answer	726	2111

QANet		
Pred/Truth	Answer	No Answer
Answer	2192	1120
No Answer	656	1983

QANet, ~~convs~~ → attn		
Pred/Truth	Answer	No Answer
Answer	41	37
No Answer	2807	3066

(dev)	**Overall**	**Who**	**What**	**When**	**Where**	**Why**	**How**	**Misc.**
Count	5951	601	2759	440	231	84	525	1311
EM	63.57	64.39	62.78	**70.68**	61.04	58.33	60.19	60.18
F1	66.42	66.71	66.28	**72.12**	67.03	64.11	63.67	64.46
AvNA	71.55	70.72	71.04	**76.82**	73.59	71.43	68.00	70.40
AvNA TPR = 74.51		AvNA TNR = 68.03		AvNA FPR = 31.97		AvNA FNR = 25.49		
EM	61.27	61.06	59.70	**67.73**	60.17	52.38	60.95	58.50
F1	64.49	64.06	63.72	**69.60**	66.89	57.35	65.77	62.83
AvNA	70.16	68.72	69.52	**74.55**	73.16	70.24	70.48	70.02
AvNA TPR = 76.97		AvNA TNR = 63.91		AvNA FPR = 36.09		AvNA FNR = 23.03		
EM	51.87	52.41	53.75	43.18	49.35	47.62	**54.29**	50.11
F1	51.89	52.41	53.75	43.44	49.35	47.62	**54.29**	50.11
AvNA	52.21	52.41	53.75	48.18	49.35	47.62	**54.29**	50.19
AvNA TPR = 1.44		AvNA TNR = 98.81		AvNA FPR = 1.19		AvNA FNR = 98.56		

Table 3: AvNA data and EM/F1/AvNA scores by question type for Ensembled QANet, Baseline QANet, and QANet with convolutions replaced with global attention.

Because the original QANet paper hypothesized that the convolutions layer primarily served to capture local structure of the text, we were interested to see what would happen if we swapped out the convolutions layer for more self-attention (Corresponding to Table 2: "QANet, replace conv with global self-attention"). Since traditional self-attention is done with respect to the entire input (i.e., it is "global"), we expect such a model to perform worse than basic QANet, as this variation's ability to capture local structure is being diluted with less relevant far-away observations. When implementing this model, we found that having too many layers of global attention would result in memory problems, so this model used as many additional self-attention layers as memory-permitting: 2 extra layers in the embedding encoder and 1 extra layer in the model encoder. Each extra layer of self-attention was followed by a feed-forward layer. The resulting model performed poorly as expected, but notably, it did outperform the version of QANet with no convolutions, meaning that the use of a repeated layer that poorly captures local attention does lead to better performance than skipping the repeated layer altogether.

The next step, then, was to build a version of the model that used a more local self-attention scheme as a replacement to the convolution layer (Listed in Table 2 as "QANet, replace conv with local self-attention"). These local self-attention layers were limited to considering a window roughly 25% the length of the entire input. This constraint eliminated the memory problems present in the global self-attention, so we were able to use the same number of local self-attention layers as the number of convolution layers in the original model (4 layers in the embedding encoder and 2 layers in the model encoder). No additional feed-forward layers were used in this model. Results showed that this model did outperform the global self-attention model described above, which is consistent with the original authors' hypothesis, although results for this model were still a far cry from the basic QANet. We have two possible explanations for this. First, our arbitrarily chosen 25% window size might not have been reflective of the local scope at which the convolution later operates—more experimentation with different window sizes for the substitutive local self-attention may further reveal insights. Second, it is not exactly clear how "powerful" a single layer of local self-attention is in comparison to a single convolution layer. How many layers of local self-attention are required to match the expressive power of a single convolution layer? This model was built on the assumption that the two layers have roughly equal "brain power," which may be erroneous. Further analysis with differing numbers of local self-attention layers to substitute a single convolution layer may yield a result more comparable to that of the QANet baseline.

In Table 3, we analyze the AvNA data and accuracy by question type (Who, What, Where, When, etc.) for three models: the ensembled QANet model, the basic QANet model, and the QANet model where convolutions were replaced with global self-attention. Both the QANet and QANet ensemble

models perform best on **"when"** questions. We believe that this is because there is a limited number of valid answers to "when" questions; there are only so many ways to represent a date or a time. This is in stark contrast to **"why"** questions, the type of questions these models generally perform worst on, as "why" questions are inherently much more complex (e.g., needing to infer intent) and have answers that can look very different depending on the input and may not have clearly distinguishable landmarks, dates, or times in the answer text. By contrast, the poorly performing model that replaces convolutions with global self-attention performs best on **"how"** questions. As can be seen from its AvNA results, this model is very prone to predicting "No Answer", so we believe that this model performs best in the "how" category because of there being relatively few answerable questions of this category, not because the model actually has a better understanding of "how" questions.

An analysis of the AvNA values for the QANet and QANet ensemble models reveals that these models would improve by more effectively learning when to give an answer and when to abstain. Both models have false negative rates in the 20s and false positive rates in the 30s. The QANet ensemble AvNA values demonstrate that this model is better at correctly refraining than the individual 7-block QANet model (compare 68.03% vs 63.91% true negative rates), likely because the constituent ensemble ingredients were more likely to abstain. Perhaps we would see further performance improvements with the introduction of another layer into the QANet architecture specifically designed to assess if a given question is answerable or not given the context.

Finally, we explored some individual questions and notice three patterns in errors:

- **The prediction occasionally captures extraneous words.** For instance, a prediction is "proteolysis into peptides" when the answer is simply "proteolysis". This suggests the model can correctly identify the context words that align with the meaning of the question, but may struggle with precisely understand the question to capture only what is needed.
- **The prediction is often No Answer when the true answer has complex grammatical structure.** For instance, a prediction of No Answer was generated for correct answer "leftist/communist/nationalist insurgents/opposition". This is likely because the training set does not have such an unusual answer format and is thus difficult to learn.
- **If the question has a suggestive word (e.g., "religious" as in "What general religious belief did the nations that received Huguenot refugees have in common?"), the model tends to generate an incorrect answer (e.g., "Protestant") when the correct choice is No Answer.** This suggests the model might be overly influenced by key words in the question and should instead prioritize gaining a higher-level understanding of its meaning.

7 Conclusion

Through our experiments re-implementing QANet, we achieved our best performance on an ensemble model consisting of several QANets with differing numbers of encoder blocks (+8.4 F1 and +8.6 EM compared to BiDAF baseline). We also achieved an understanding that corroborates the original paper's hypothesis that the convolution block predominantly captures local structure (more so than big-picture global interaction), since our results indicate improved performance when replacing convolution with local self-attention relative to global self-attention.

A key limitation is that our model does not condition the end probability on the start position probability, since practically, the end position depends on the start. Future work could modify the output layer to capture this relationship. Another limitation is that our replacement of convolution with local self-attention assumed that there was a one-to-one correspondence in representational power. To more deeply understand the role of QANet's convolution layer, future work should determine how to equate a number of convolution layers with local attention layers, along with further analysis about the most appropriate window size for convolution-imitation local self-attention layers. To improve QANet performance, further experiments should be conducted to investigate how changes in stochastic depth dropout rate impact performance and if performance can be boosted by modifying the number of layers in the embedding encoder. Finally, the original QANet paper also experimented with data augmentation by using machine back-translation to generate various wordings of original text; experimenting with more languages may yield a more syntactically diverse dataset and improve performance.

References

[1] 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. In *arXiv preprint arXiv:1804.09541*, 2018.

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

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

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

[5] Zhuang Liu Daniel Sedra Kilian Q. Weinberger Gao Huang, Yu Sun. Deep networks with stochastic depth. In *arXiv preprint arXiv:1603.09382*, 2016.