# BiDAF QAN Be Just As Good: A Comparison Between Improved BiDAF and QANet

Stanford CS224N IID SQuAD Default Project

**Jerry Liu**
Department of Computer Science
Stanford University
jerrylzy@stanford.edu

**Yuke Wu**
Department of Computer Science
Stanford University
yukewu@stanford.edu

## Abstract

In this project, our goal is to build a question answering model that works well on the Stanford Question Answering Dataset (SQuAD) 2.0. To achieve this goal, we experiment with various modifications to the BiDAF model and try to implement QANet from scratch. We adopt three techniques to improve the baseline BiDAF implementation: char-level embedding, self-attention and POS tagging. We find that with the help of character embedding and self-attention, BiDAF manages to achieve a remarkable performance – an EM score of 64.8 and an F1 score of 68, comparable to our best performing QANet implementation. Using an ensemble of BiDAF and QANet models, we achieve an EM score of 69.47 and an F1 score of 71.96, ranking #3 on the leaderboard as of March $14^{\text{th}}$.

## 1 Key Information to include

- Mentor: Christopher Wolff
- External Collaborators (if you have any): N/A
- Sharing Project: N/A

## 2 Introduction

Question answering is a fundamental natural language understanding problem and a long-standing artificial intelligence milestone. It is an important NLP problem because of its widespread usage in practical applications and its ability to demonstrate reading comprehension — something many other NLP tasks reduce to. A typical QA system locates the correct answer from the provided context and presents it in the form of a natural language answer to the given question. SQuAD 2.0 is a question answering dataset that contains questions whose answers are not stated in the contexts, extending beyond the scope of traditional QA systems. This is a challenging task for many existing QA systems. The goal of this project is to create a question answering model that works well on both the difficult no-answer questions and the traditional answerable contextual questions from SQuAD 2.0.

The Bidirectional Attention Flow network (BiDAF) [1] is the baseline model of this project. BiDAF is a QA architecture that leverages an innovative bi-directional attention mechanism to obtain query-aware context representation. To improve the BiDAF baseline, we adopt three techniques: character-level embedding, self-attention, and POS tagging. Character-level embedding applies one-dimensional convolutional neural networks to obtain word vectors by looking at their character-level compositions. It better represents out-of-vocabulary words since it conditions on morphology. Because BiDAF's word embedding layer (GloVe) represents out-of-vocabulary words by simply assigning random vector values, we expect incorporating character-level embedding to improve model performance by providing better language representation. We also implement the self-attention mechanism [2] to address BiDAF's weakness in capturing long-term dependencies. Self-attention includes contextual relevance by relating positional information and comparing every word in the

sequence to every other word. Lastly, we experiment with part-of-speech (POS) tagging within the embedding layer. We expect that taking the syntactic information of input vectors into account facilitates the use of linguistic criteria to improve the model.

QANet [3] was developed when reading comprehension was dominated by RNN-based models including BiDAF [1]. RNN-based models achieved great performance in question answering due to their natural way of handling sequential text inputs. However, RNN-based models are difficult to parallelize and struggle to capture long dependencies even with the emergence of Gated Recurrent Units (GRU) [4] and Long Short-Term Memory (LSTM) [5], leaving room for improvement in speed and accuracy. Because of the bottlenecks stemming from the recurrent nature of most reading comprehension models, we implement QANet [3], a new end-to-end model inspired by the Transformer architecture, with the core aim of overcoming the inherent shortcomings of RNN-based reading comprehension models.

## 3 Related Work

A question answering system extracts the answer to a query from a given context, which requires extensive interactions between context and query. BiDAF introduces a mechanism that allows attention to flow from query to context and from context to query, achieving state-of-the-art performance on SQuAD at the time of publication. However, as an RNN model, its inherent difficulty with parallelization and capturing long-term dependencies led to the popularization of the Transformer [2] and Transformer-based attention networks [3]. The Transformer architecture shifted away from recurrence and convolutions entirely and is solely based on attention mechanisms. It achieved 28.4 BLEU on the WMT 2014 English-to-German translation task, significantly improving on the existing best results at the time.

Yu et al.'s QANet model replaced all recurrent units with parallelizable convolutions and self-attentions. The model achieved a 3x to 13x speedup in training and 4x to 9x speedup in inference when compared to its RNN counterparts due to its feed-forward nature. This improvement in speed led to great performance on SQuaD (84.6 in F1) through the introduction of a larger dataset, more training iterations, and data augmentation techniques like backtranslation.

## 4 Approach

### 4.1 Improve BiDAF

#### 4.1.1 Character-Level Embedding

To obtain the input embeddings, we combine the pre-trained $d = 300$ GloVe word-level embeddings — which are frozen during training — [6] with a trainable character embedding layer. Each character is represented by a trainable $d = 64$ or $d = 200$ vector. Shorter words are padded and longer words are truncated to 16 characters. We feed the concatenated embeddings into the linear projection layer which then outputs to the two-way highway encoder [7]. We implement an option to enable a character CNN layer with a kernel size of 5 which is followed by a maxpool layer that outputs the highest activation for each embedding dimension out of all 16 characters. The final output of the character CNN has the same dimension as the hidden size of a chosen model. Droput is applied right before the linear projection layer.

#### 4.1.2 Self-Attention

The self-attention layer, illustrated by the right side of Figure 1, is composed of a sinusoidal positional input encoding layer, a multi-head attention layer, and a feed-forward layer [2]. We take the output of the positional embedding layer to compute queries, keys, and values. We then calculate the normalized attention scores between queries and keys and multiply them by their corresponding values. Each basic layer is accompanied by layer normalization and residual dropout to avoid issues such as lack of element-wise non-linearities and distorted information. The self-attention output is concatenated with the Context2Query and Query2Context attentions within the attention flow layer, as shown in Figure 1.

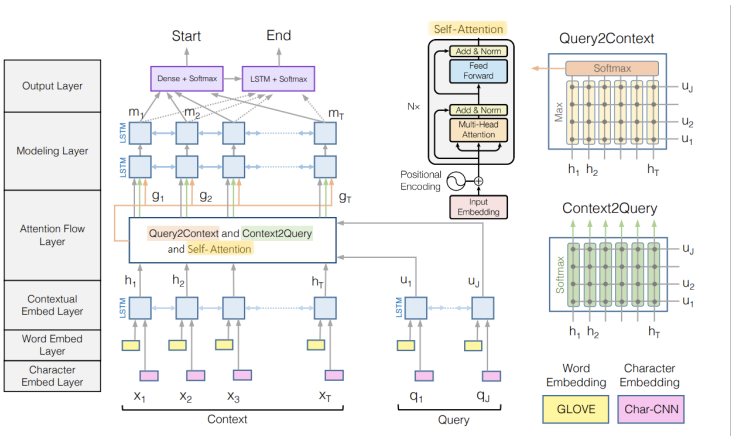


Figure 1: BiDAF with Self-Attention

### 4.1.3 Part-of-Speech Tagging

We use NLTK's averaged perceptron tagger to generate POS tags for all input words. The tags are transformed into one-hot vectors and concatenated to the word and character embeddings, which are then fed into the linear projection and highway encoding layers to generate model embeddings.

## 4.2 Implement QANet

QANet consists of five major layers: embedding, embedding encoder, context-query attention, model encoder and output [3], as shown in Figure 2.

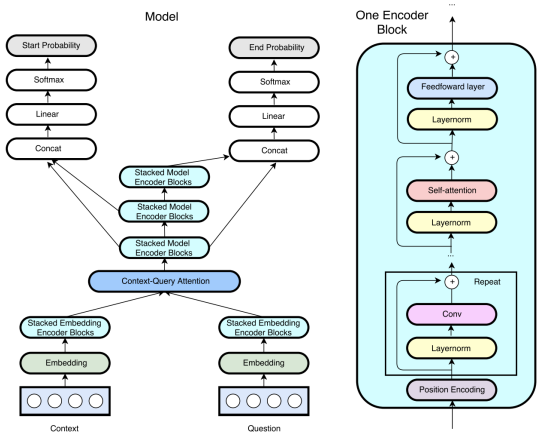


Figure 2: The QANet architecture.

### 4.2.1 Input embedding layer

The input layer is similar to the BiDAF input layer described in Section 4.1.1. Our implementation has a separate dropout for word embeddings and character embeddings, with a rate of 0.1 and 0.05 respectively. We use 1-D convolution of size 1 instead of a linear layer to match the original QANet implementation.

### 4.2.2 Embedding encoder layer

The encoder block, illustrated by 1, is composed of a fixed sinusoidal positional input encoding layer adopted from [2], multiple depth-wise separable convolutions [8], a multi-head attention, and a feed-forward layer. Each layer other than the positional encoder is wrapped inside a *residual block* $f(\text{layernorm}(x) + x)$ where we apply layer normalization to the input and add a residual connection

to allow an identity path. To process the input embedding, we apply a single encoder block containing 4 convolutions with a kernel size of 7 and an 8-head attention layer.

#### 4.2.3 Context-query attention layer

This layer is directly adopted from BiDAF [1]: we compute the similarities between each pair of context and query words, generating Context-to-Query attention and Query-to-Context attention.

#### 4.2.4 Model encoder layer

Each model encoder has 7 encoder blocks identical to the model encoder block, except each block contains 2 convolutions with a kernel size of 5. We share weights amongst the 3 repetitions of the model encoder.

#### 4.2.5 Output layer

This layer predicts the probability of start and end positions of an answer in the given context. The probabilities of the start and end positions of the predicted answer are

$$p_1 = \text{softmax}(W_1[M_0; M_1]) \qquad\qquad p_2 = \text{softmax}(W_2[M_0; M_1])$$

where $M_0, M_1, M_2$ are the outputs of the 3 model encoders from the previous layer and $W_1, W_2$ are trainable weights. Assuming $l$ is the length of the context, the objective function takes the predicted probabilities and computes the negative averaged sum of the log probabilities of the predicted distributions indexed by true start and end indices:

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

where $\theta$ represents all trainable variables and $y^1_i, y^2_i$ are the ground truth start and end positions of example $i = 0, \ldots, l-1$.

The predicted answer $(s, e)$ is chosen from

$$(s, e) = \underset{s \le e}{\text{argmax}} \; p^1_s p^2_e$$

where $s = e = 0$ indicates no answer.

#### 4.2.6 Ensemble

We combine both self-attention BiDAF and QANet variants to explore any performance improvement from ensemble modeling. To obtain the ensemble model output, we calculate the maximum pair probability. For a given model $m_i, i = 1, \ldots, n$, $p_{s_i}, p_{e_i} \in \mathbb{R}^l$ are the probability vectors for the start and the end positions where $l$ is the length of the context.
We constructed the probability matrix for all pairs of positions as

$$P_i = p_{s_i} p^\top_{e_i}$$

Let $P_{i_{j,k}}$ be the $(j, k)$ entry of matrix $P_i$, we define

$$P_{j,k} = \max_{1 \le i \le n} P_{i_{j,k}}$$

Then, we pick the start and end position $s, e$ to be

$$s, e = \underset{0 \le i < j < l}{\text{argmax}} P_{i,j}$$

This ensures the selection of the most confident answer (the pair with the highest joint probability) from all models. Note that $s = e = 0$ indicates no-answer.

## 5 Experiments

### 5.1 Data

We use the Stanford Question Answering Dataset (SQuAD) 2.0 [9] for training and testing the question answering system. The SQuAD 2.0 dataset used for this project contains 129,941 question-context-answer tuples for training, 6,078 for dev/validation, and 5,915 for testing. Compared to the

original SQuAD, SQuAD 2.0 integrates over 50,000 adversarial questions whose answers are not stated in their contexts. This ensures that the output model is able to locate the correct answer within the context documents while being more robust against unanswerable questions.

### 5.2 Evaluation method

To measure model accuracy, we use Exact Match and F1 scores as our evaluation metrics. We compare our model scores to the BiDAF baseline — 55/58 in EM/F1 on the dev dataset. We also evaluate our QANet implementation against Back et al.'s EM/F1 scores of 63.6/66.7 on the dev set. In addtion to EM/F1, we also report the answer vs. no-answer (AvNA) scores for reference.

### 5.3 Experimental details

For all the models, we take advantage of Automatic Mixed Precision (AMP) from NVIDIA CUDA for training. It reduces our training time by 18% for QANet and 30% for BiDAF. Models are trained on a single Tesla V100 with 6 Xeon vCPUs. For all models, we use Kaiming normal initialization [11] for Conv layers with ReLU activations and Xavier uniform initialization [12] for conv layers without ReLU activation. For all other layers, we use the PyTorch default initialization, which is usually $\mathcal{N}(0, 1)$.

#### 5.3.1 BiDAF

The BiDAF model uses the starter code's parameters – it has a hidden size of 100, a batch size of 64, an EMA decay of 0.999, and a constant learning rate of 0.5. It uses the AdaDelta [13] optimizer without L2 regularization. A dropout [14] rate of 0.2 is applied for the concatenated word and character embedding, all LSTM layers, all sublayers of the Self-Attention layer if applicable, and the final linear layer before outputting the answers with a masked softmax function. All RNN parameters are flattened to optimize performance on a CUDA GPU. The training takes about 2.5 hours for 30 epochs on a single Tesla V100 GPU.

#### 5.3.2 QANet

Like what Yu et al. did in their original implementation, we use both L2 weight decay and dropout for regularization. The L2 weight decay is $\lambda = 3 \times 10^{-7}$. We also use a dropout rate of 0.1 for word embeddings and 0.05 for character embeddings. We also use a dropout rate of 0.1 between every two layers. We also incorporate the stochastic depth method [15] within the embedding and model encoder layers. Each sublayer $l$ has a dropout probability of $p_l = \frac{l}{L} p_L$, where $L$ is the last layer and $p_L = 0.1$.

The hidden size and the number of filters for all convolutions are both 128. The batch size is 32. There are 4 depth-wise separable convolutions with a kernel size of 7 in the embedding layer and 2 depth-wise separable convolutions with a kernel size of 5 in the modeling encoder layer. There is 1 encoder block for the embedding encoder and 7 stacked encoder blocks for the modeling encoder layer.

As in the original QANet paper, we use the Adam optimizer [16] with $\beta_1 = 0.8, \beta_2 = 0.999$ and $\epsilon = 10^{-7}$. We also implement learning rate exponential warm-up before the 1000th step and keep it constant at 0.001 for the rest of training. We also use an EMA decay of 0.9999. The training takes about 8.5 hours for 30 epochs on a single Tesla V100 GPU.

#### 5.3.3 Ensemble Model

We constructed two ensemble models with the maximum pair-wise probability method mentioned in Section 4.2.6: 1. Ensemble-5: 3 self-attention BiDAF models and 2 QANet models where one model uses character CNN and the other does not; and 2. Ensemble-7: the same configuration as Ensemble-5 with the addition of one self-attention BiDAF model and one QANet model without character CNN.

### 5.4 Results

The performance results of our models on the SQuAD 2.0 dataset (IID SQuAD track) are shown in Table 1. Character-level embedding and self-attention both lead to significant performance

improvement. Increasing the character embedding size to 200 achieves a very high performance on the dev set, comparable to QANet models. This demonstrates the importance of self-attention's role in transformer models.

However, adding a single-layer CNN on top of character embedding, described in both BiDAF and QANet [1, 3], degrades performance on both model types. We additionally find that using the usual $\mathcal{N}(0, 0.02)$ from [17] works in our QANet implementation. According to Yu et al., QANet should be

| Model | | Dev Set | | | Test Set | |
|---|---|---|---|---|---|---|
| | | EM | F1 | AvNA | EM | F1 |
| BiDAF | Baseline | 57.70 | 61.25 | 68.02 | | |
| | 64D Char Embedding (C-Emb) | 60.71 | 64.26 | 70.63 | | |
| | 64D C-Emb, Self-Attention | 61.17 | 64.38 | 70.51 | | |
| | 200D C-Emb, Self-Attention | **64.80** | 68.00 | 73.75 | 62.01 | 65.8 |
| | 200D C-Emb, Self-Attn, Char CNN | 63.01 | 66.07 | 71.80 | | |
| | 200D C-Emb, Self-Attn, POS Tag | 64.16 | 67.11 | 73.25 | | |
| QANet | Baseline from [10] | 63.60 | 66.70 | | | |
| | 64D C-Emb, Baseline | 62.53 | 66.52 | 73.28 | | |
| | 200D C-Emb, Baseline | 64.02 | **68.14** | **74.79** | | |
| | 200D C-Emb, Stochastic Depth | 64.43 | 68.00 | 74.02 | | |
| | 200D C-Emb, Char CNN | 63.75 | 67.28 | 73.53 | | |
| Ensemble-4 | 4 BiDAF | 68.14 | 70.67 | 75.45 | | |
| Ensemble-5 | 3 BiDAF + 2 QANet | 69.05 | 71.78 | 76.63 | | |
| Ensemble-7 | 4 BiDAF + 3 QANet | **69.47** | **71.96** | **76.64** | **66.12** | **68.88** |

Table 1: Performance on the SQuAD 2.0 dev and test sets.

3x to 13x faster than an RNN-based model like BiDAF. However, we find that even a complex BiDAF model trains up to 4x faster than QANet. Some possible causes include: 1. the RNN implementation used by Yu et al. is extremely inefficient and 2. the QANet implementation on Tensorflow is much more efficient. Batch size does play a role in training speed, but the original authors use the same batch size of 32.

From the training loss curve shown in Figure 6, we can see that QANet learns much faster during initial epochs. However, QANet's training NLL curve plateaus while BiDAF continues to learn from and overfit the training data. We find that using *StepLR* or *CyclicLR* to vary or decrease the learning rate, typically used to solve the pleateauing issue, does not improve performance of our QANet model.

We find that ensemble models significantly improve performance. Our 7-model ensemble achieves 69.47 in EM and 71.96 in F1 on the dev set and 66.12 in EM and 68.88 on the test set. Even a BiDAF-only ensemble mode achieves remarkable performance uplift over our best-performing BiDAF model, reaching an EM/F1 score of 68.14/70.67. This demonstrates the power of model ensembling and makes the case that training multiple simpler models can be more beneficial than a more complex, slow-to-train model under a fixed budget or time constraint.

## 6 Analysis

### 6.1 Performance on Different Question Types

As we can see from Figure 3, BiDAF and QANet perform differently on different question types. We can see that while BiDAF struggles with "how" and "which" questions, especially in no-answer samples, while QANet struggles with "why" questions. The same phenomenon mostly can also be seen in the F1 and AvNA score comparisons shown in Figure 7, 8 in the Appendix as well.

We also notice there is a small but noticeable drop in performance between the dev set and the test set. Therefore, we analyze the distribution of question types, shown in Table 2. As we can see from Figure 3 and Table 2, the percentages of question categories on which our models do well fall noticeably while for the test set, our models need to answer more questions on which they do not perform well. This could explain why our models have a small but noticeable drop in performance on the dev set.

Because we tune our model solely on the performance of the dev set, our model could be overfitting the dev set. We may need to incorporate an independent, second dev set to really validate our model's performance on unseen data.

| | **Dev (%)** | Dev (#) | **Test (%)** | Test (#) |
|---|---|---|---|---|
| **How many** | **4.48%** | 272 | **5.95%** | 352 |
| **How** | **5.48%** | 333 | **6.81%** | 403 |
| **What** | **61.06%** | 3711 | **63.79%** | 3773 |
| Why | 1.5% | 91 | 1.78% | 105 |
| Which | 3.97% | 241 | 4.63% | 274 |
| **Who** | **11.39%** | 692 | **7.37%** | 436 |
| Where | 4.11% | 250 | 3.85% | 228 |
| **When** | **7.06%** | 429 | **4.97%** | 294 |
| Other | 0.97% | 59 | 0.85% | 50 |

Table 2: The distribution of types of questions. The types of questions that had noticeable change of distribution are boldfaced.

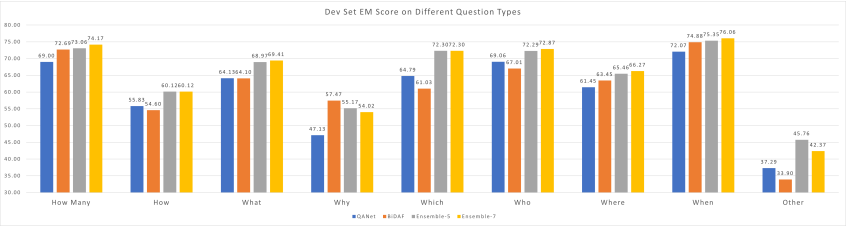


Figure 3: Dev Exact Match comparison based on question category. **Blue was QANet. Orange was BiDAF. Gray was Ensemble-5. Yellow was Ensemble-7.**

### 6.2 Investigate Performance on Certain Question Types

**"How" Questions**. We notice that our BiDAF model struggles with "how" questions. Therefore, we dig into individual questions to investigate.

We can see that our BiDAF model chose this answer because "start"'s meaning is close to "began", not because it understood the question or paid attention to the word "peace" according to the C2Q Attention map shown in Figure 4. Here, we can see that even though the QANet model paid a lot of attention to the word "began" as well, it also paid attention to other related words like "fought". This may explain why QANet performs better than BiDAF on "how" questions.

**Question:** How did peace start?
**Context:** The war was fought primarily along the frontiers between New France and the British colonies, from Virginia in the South to Nova Scotia in the North. It began with a dispute over control of the confluence of the Allegheny and Monongahela rivers, called the Forks of the Ohio, and the site of the French Fort Duquesne and present-day Pittsburgh, Pennsylvania. The dispute erupted into violence in the Battle of Jumonville Glen in May 1754, during which Virginia militiamen under the command of 22-year-old George Washington ambushed a French patrol.
**Answer:** N/A
**Prediction:** with a dispute over control of the confluence of the Allegheny and Monongahela rivers

There is another example of "how" question in Section A.2.1 where the QANet produces the correct answer and pays more attention to relevant words in the context. Because the C2Q attention layer is the same, this indicates that QANet's **self-attention + convolution** embedding encoder performs better than a bi-LSTM.

**"Why" Questions**. The following question is an example of "why" question where QANet does not produce the right answer. According to the attention map in Figure5, we can see that QANet produces better C2Q attention because on the row "why" it paid attention to relevant texts in the context. This may be an indication that our model output layer for QANet is too simple because while QANet outputs start and end positions independently, BiDAF actually feeds the output of the start position to be an input to the modules calculating the end position.

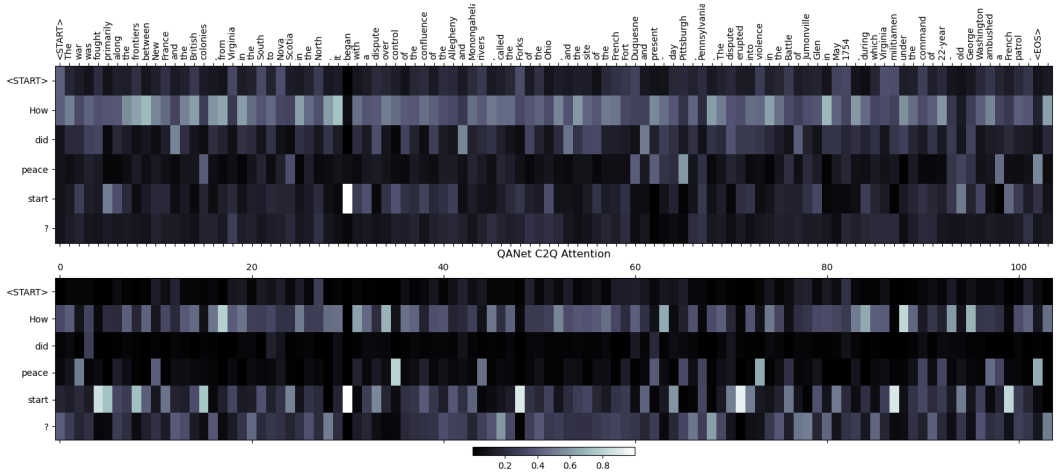


Figure 4: Attention Map of C2Q Attention for this question.

**Question:** Why did French feel they had right to Ohio claim?
**Context:** Jacques Legardeur de Saint-Pierre, who succeeded Marin as commander of the French forces after the latter died on October 29, invited Washington to dine with him. Over dinner, Washington presented Saint-Pierre with the letter from Dinwiddie demanding an immediate French withdrawal from the Ohio Country. Saint-Pierre said, "As to the Summons you send me to retire, I do not think myself obliged to obey it." He told Washington that France's claim to the region was superior to that of the British. since René-Robert Cavelier, Sieur de La Salle had explored the Ohio Country nearly a century earlier.
**Answer:** France's claim to the region was superior to that of the British
**BiDAF Prediction:** France's claim to the region was superior to that of the British
**QANet Prediction:** N/A

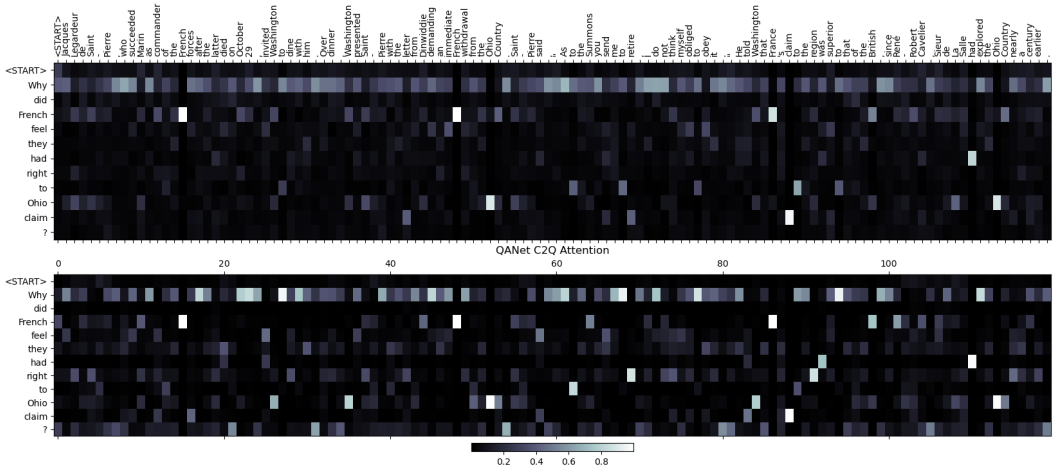


Figure 5: Attention Map of C2Q Attention for this question.

## 7 Conclusion

In conclusion, we develop a powerful implementation of BiDAF and a better-than-reference QANet [10] that provide strong results on the SQuAD 2.0 dataset. We show that parts-of-speech tagging using one-hot vectors, single-layer character CNN, stochastic depth, and adjustable learning rate do not improve model performance. Character-level embedding and self-attention, on the other hand, both lead to non-trivial performance improvements for BiDAF. The ensemble models consisting of BiDAF and QANet models achieve much stronger performance on SQuAD 2.0. With a 7-model ensemble, we achieve a **top-3** finish on the dev set leaderboard as of March 14$^{th}$, with EM/F1 scores of 69.47 and 71.96. As for training time, we notice that QANet is much slower to train and consumes

much more memory than BiDAF on today's software and hardware. Given the stark difference in training time between BiDAF and QANet and little difference in performance, we find that training more models that are architecturally simpler can lead to better performance than experimenting with improvements on a single more complex and slow-to-train model under a fixed time constraint. We also find that QANet's self-attention + convolution embedding encoder works better than BiDAF's bi-LSTM from our attention analysis.

The primary limitation of our work is that the input embedding layer becomes the bottleneck in our BiDAF and QANet implementations and can be further explored with other techniques. In the future, we would like to mimic POS tags as characters to use a learnable small-dimension embedding. We would also like to incorporate more complex character CNNs to see if it leads to any performance improvement. As the deep learning world trends towards large pre-trained transformer models, we would like to see if adopting a BERT-like encoding layer can significantly improve performance on SQuAD 2.0.

## References


[1] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HJ0UKP9ge.

[2] 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.

[3] Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B14TlG-RW.

[4] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[6] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

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

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

[9] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018. URL http://arxiv.org/abs/1806.03822.

[10] Seohyun Back, Sai Chetan Chinthakindi, Akhil Kedia, Haejun Lee, and Jaegul Choo. Neurquri: Neural question requirement inspector for answerability prediction in machine reading comprehension. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxgsCVYPr.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. URL http://arxiv.org/abs/1502.01852.

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[13] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL `http://arxiv.org/abs/1212.5701`.

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[15] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. *ECCV*, abs/1603.09382, 2016. URL `https://arxiv.org/abs/1603.09382`.

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

# A Appendix

## A.1 Figures

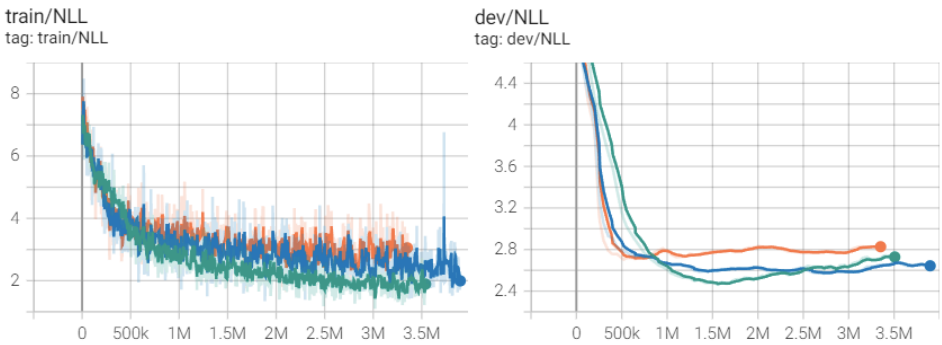


Figure 6: Train and Dev Negative Log-Likelihood Loss (NLL). **Green line: BiDAF. Orange line: QANet. Blue line: QANet with Stochastic Depth.** We can see that QANet's training loss plateaued after 10-15 epochs.

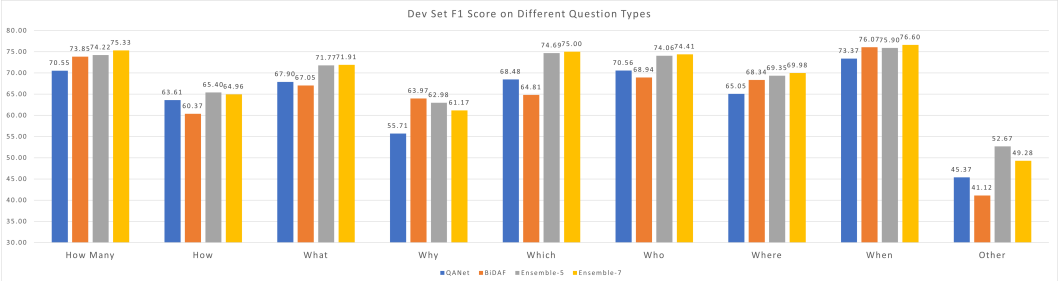


Figure 7: Dev F1 comparison based on question category. **Blue was QANet. Orange was BiDAF. Gray was Ensemble-5. Yellow was Ensemble-7.**

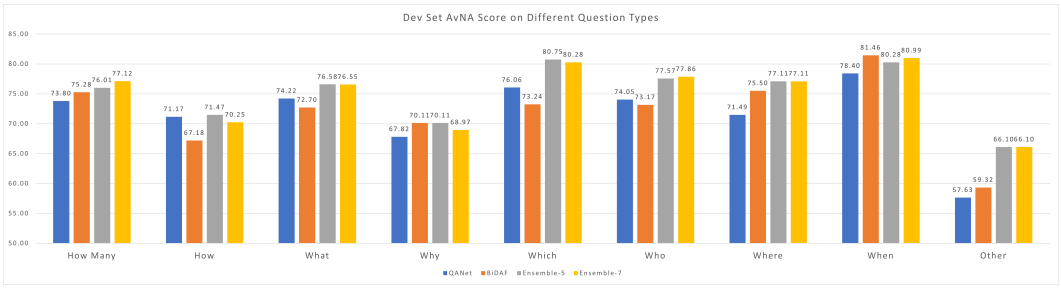


Figure 8: Dev AvNA comparison based on question category. **Blue was QANet.** **Orange was BiDAF.** **Gray was Ensemble-5.** **Yellow was Ensemble-7.**

## A.2 Attention Analysis

### A.2.1 Another "How" Question

As we can see, in this case, QANet's C2Q attention paid a lot more to the word few for the words "much" and "?" in the question, shown in Figure 9.

**Question:** How much British military was in North America at start of War?
**Context:** At the start of the war, no French regular army troops were stationed in North America, and few British troops. New France was defended by about 3,000 troupes de la marine, companies of colonial regulars (some of whom had significant woodland combat experience). The colonial government recruited militia support when needed. Most British colonies mustered local militia companies, generally ill trained and available only for short periods, to deal with native threats, but did not have any standing forces.
**Answer:** few British troops, few
**BiDAF Prediction:** no French regular army troops were stationed in North America
**QANet Prediction:** few

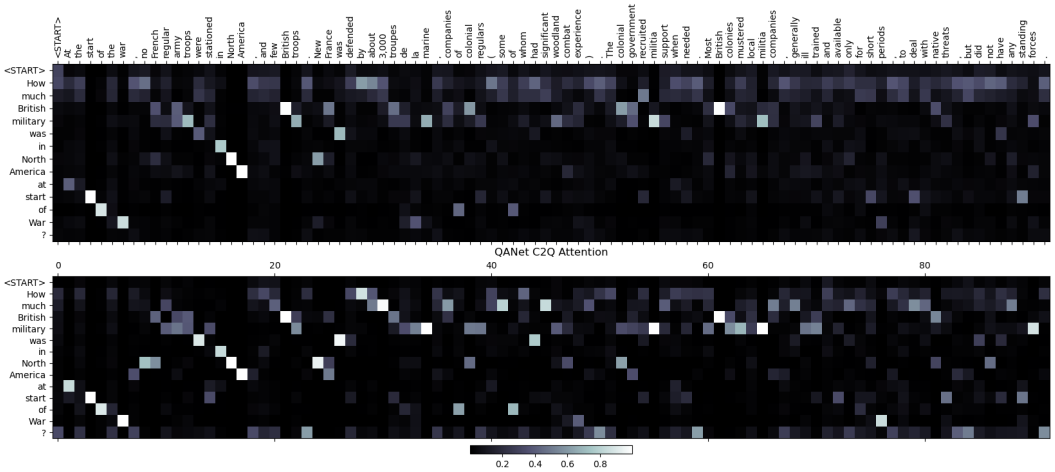


Figure 9: Attention Map of C2Q Attention for this question.

## A.3 Acknowledgement

We adopted the Depthwise Separable Convolution implementation from [1]. We adopted some initialization techniques from an open-source QANet model [2]. We used PyTorch's positional encoder

[1] https://github.com/seungjunlee96/Depthwise-Separable-Convolution_Pytorch/blob/master/DepthwiseSeparableConvolution/DepthwiseSeparableConvolution.py

[2] https://github.com/BangLiu/QANet-PyTorch

and MultiHeadAttention implementation [3] in QANet to find any performance difference between the PyTorch implementation and our own. We did not find any difference. However, due to the amount of time required to retrain a model, we kept the PyTorch implementation for QANet. Our BiDAF model still uses our own implementation of those two modules, which is written by Jerry for his CS 231N homework last year.

[3] https://pytorch.org/tutorials/beginner/transformer_tutorial.html