Extended QANet on SQuAD 2.0
GRADING OPTION 3

Stanford CS224N Default Project - Final Project Report

Guillaume Nervo
ICME
Stanford University
`gnervo@stanford.edu`

Pablo Veyrat
Management Science and Engineering
Stanford University
`pveyrat@stanford.edu`

Abstract

In this project, we built question answering systems for the Stanford Question Answering Dataset (SQuAD) 2.0. We explored two end-to-end models: the baseline BiDAF network ([1]), and QANet ([2]), a non-recurrent model fully based on convolution and self-attention. Two major goals were accomplished. Firstly, we improved the baseline BiDAF model by introducing character embeddings, and a linear ReLU activation in the fusion function of the attention layer. Secondly, we re-implemented the QANet model from scratch and successfully explored some variations on its architecture. Our best BiDAF/QANet single model achieved $62.46/64.54$ EM and $65.76/68.43$ F1 score on the development set respectively. We also built an ensemble model which achieved $\boldsymbol{EM = 69.16}$ and $\boldsymbol{F1 = 71.77}$ on the development set, and $\boldsymbol{EM = 66.32}$ and $\boldsymbol{F1 = 68.94}$ on the test set.

1 Introduction

Reading comprehension is the ability to process text and understand its meaning. In this project, we will be performing question answering, which is one of the most studied machine comprehension task. Recently a variety of neural architectures achieved near-human performance in open domain question answering tasks. These recent advances are broadly of two types: (1) Pre-trained Contextual Embeddings (PCE) based methods such as ELMo [3] and Bert [4] and (2) Non-PCE methods. The former offers more of an "off-the-shelf" module that could be employed for specific tasks such as question answering, when the latter, even though not being the state of the art (SoTA) any longer (as of March 2020), offers more scope for creativity and opportunities for deep learning practitioners to explore different techniques and develop intuitions behind them.

In this work, we have experimented with some of the most successful non-PCE approaches to question answering like the QANet model, with the overall goal to achieve the highest performance possible on the non-PCE SQuAD 2.0 leaderboard. QANet, which was built by Yu et al. in [2] for speed and efficiency, achieved state-of-the performance on SQuAD 1.0. However, suitability of the architecture for the non-answerability setting has not been systematically evaluated to our knowledge. Our major focus in this project has been implementing the QANet model, understanding its limitations against the SQuAD 2.0 dataset and investigating methods to augment its architecture.

We have also worked on the baseline BiDAF model ([1]), and showed that we could improve performance by adding a character embedding layer and by exploring a new fusion function in the bi-directional attention layer. We have ensembled our BiDAF models with our top-performing QANet models. These models being significantly diverse, the ensemble have yielded better results than the single models. Last, we have analyzed the outputs of our models which helped us better understand how they work, in which situations some models produce relevant or irrelevant outputs, which layers are most important, and how we could further improve them in a future work.

2 Related Work

Prior to QANet, major question answering systems primarily contained either of two key ingredients (1) recurrent units such as LSTM for capturing sequential input and (2) exploiting attention mechanism for capturing long-term interactions.

The BiDAF [1] model which employs both the techniques, is a hierarchical multi-stage end-to-end network which takes inputs of different granularity (character, word and phrase) to obtain a query-aware context representation using memory-less context-to-query (C2Q) and query-to-context (Q2C) attention. This representation can then be used for different final tasks, such as question answering. This model performed particularly well for its time, as an ensemble model based around BiDAF was able to outperform all previous approaches at the time the paper was published back in 2016. Note that this paper targets SQuAD 1.0, which only contains answerable questions and therefore constitutes an easier task than what we are trying to perform here on SQuAD 2.0. One major disadvantage of this model is that it is heavily reliant on RNNs, and therefore difficult to parallelize. This in turn means that the model is slow to train: it does not take enough advantage of parallel computing.

To get rid of this non-recurrent architecture and achieve a speedup, one-and-a-half years later, the QANet [2] was published. It borrowed neat ideas from NMT proposed in the Transformer [5] architecture: encode the question and context separately using a non-recurrent, therefore faster, encoder. This encoder uses convolution (which models the local features) and self-attention (which processes global interactions) as building blocks. The authors estimated that QANet was 4.3 times faster than BiDAF to train, and 7.0 times faster to evaluate, which meant they had a massive advantage as they could use data augmentation techniques to increase the number of training examples available, and thereby process more training data than the BiDAF model. QANet achieved state-of-the-art performance when it was released, significantly outperforming the previous best model on the official leaderboard. Still, the QANet authors did not compare variations of their model with close and relevant types of model structures, with for example unshared weights for the stacks of encoders of the model encoder layer, or with different answer pointers in the output layer. In this paper, we investigate these unexplored variations on QANet while tuning the model to SQuAD 2.0.

3 Approach

The first thing we did was improve on the BiDAF model that was already implemented in the starter code for the default final project by adding character embeddings to it, like what had been done in homework 5 for the neural machine translation task using character embeddings.

The second step of our work was implementing from scratch a QANet model following the details given by Yu et al. in [2] and adapting it to the SQuAD 2.0 framework. Implementing such a model has been made easier by the fact that it shares many similar components with the BiDAF model. We thus managed to write all the components specific to this model ourselves, except for the positional encoding sub-layer for which we had to inspire ourselves from several implementations found on Github (mainly this one and this one).

Like most high-performing question answering models, the QANet model consists of five layers, as shown in Figure 7 in the appendix. The **input embedding layer** is very similar to the one used in our BiDAF implementation. It consists of the GloVe embeddings of each word concatenated with the output of a convolutional layer that processes the embeddings of each character that makes up a word up to a maximum length of 16 characters per word. The only difference is that instead of using the pre-trained character embeddings of dimension 64, we represent each character as a trainable vector of dimension 200.

After the embedding layer, the input is then passed through the **embedding encoder layer** where we first add positional encoding, since otherwise the convolutional structure of the network would lose track of that information. The rest of this block consists of several depthwise separable convolutions as introduced in [6] (which Yu et al. argue is more memory-efficient and generalizes better than convolutions), followed by a multi-head self-attention layer (adapted from [5]), and finally a feed-forward layer. This structure is the encoding block shown in Figure 7 in the Appendix.

This layer is followed by a **context-query attention layer**. It is the same as the one used by the BiDAF model, so we did not re-implement it ourselves. The attention output is expressed as

$\boldsymbol{g}_i = [\boldsymbol{c}_i; \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{b}_i] \in \mathbb{R}^{8d}$ where d is the hidden dimension, $c_1, ..., c_N$ the context input, $a_1, ...a_N$ and $b_1, ..., b_N$ the context-to-query and query-to-context attention. In [1], more than the simple concatenation, the authors explain that it would be possible to use a linear layer followed by a ReLU activation as a fusion function between the different outputs of the attention mechanism. For BiDAF, we thus also explored the following variation in the fusion function:

$$\boldsymbol{g} = ReLU(\boldsymbol{W}_{lin}[\boldsymbol{c}_i; \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{b}_i] + \boldsymbol{b}_{lin})$$

The output of the attention layer is then fed through the **model encoder layer**, which consists of several encoder blocks chained together. The number of convolutions in each block can be different from the number we had in the encoder blocks of the embedding encoder layer. This step is repeated three times to get three output matrices M_0, M_1, M_2, and the weights are shared between each repetition.

Finally, the matrices M_0, M_1, M_2 are fed through the **output layer**. The strategy to predict the answer span is the same as what is used in the BiDAF. More specifically, in this case, the probabilities of starting and ending position are modeled as follows:

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

One thing to note is that the probability distribution of the end index is computed independently of the probability distribution of the start index. In one of our experiments, we thus decided to try conditioning the distribution of the end pointer $\boldsymbol{p_{end}}$ by the distribution of the start pointer $\boldsymbol{p_{start}}$. Our idea was that it could help the network better learn how the end of an answer span relates to its beginning. To create a tangible link between the two distributions, we explored the following possibility involving a linear layer: $\boldsymbol{p_{end}} = softmax(W_3[W_2[M_0, M_2], \boldsymbol{p_{start}}])$, with $W_2 \in \mathbb{M}_{1,2 \times hidden-size}$

4 Experiments

4.1 Dataset

The dataset that we are using is a slightly modified version of the SQuAD 2.0 dataset [7], where our training data is identical, but our dev and test sets are drawn from the official dev data. The data consists of (context, question, answer) triples. Each context is drawn from Wikipedia articles, and the answer is either "No answer", or a span from the context. In the train set, approximately 33% of the questions are unanswerable, while in the dev set, only 52% are due to a range of linguistic phenomena such as negation, antonyms, entity swapping, etc.

While the dataset provides high quality questions and answers, many of them tend to be of *wh-* type. Figure 1 shows the distribution of these question types. Since pure Transformer-based architectures have been known not to do well on very long-range dependencies [5], we also take note in Figure 1 of the distribution of the context length, answer length and answer start index for the dev and the train set.

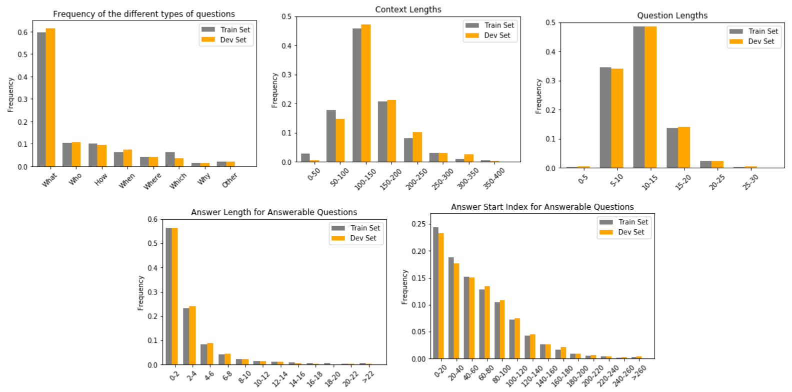


Figure 1: Dataset Distributions

We initially wanted to incorporate features from [8] in our models to better take into account such dependencies. Nonetheless, since 95% of the training contexts are of length less than 250, we understood that it would not significantly improve our models.

Curiously, when analyzing some of the contexts and questions individually, we also noted some errors in the ground truth in the dev set, especially with paragraphs dealing with highly technical topics. For example, for the question: *What is the force equivalent of torque compared to angular momentum?*, none of the human annotators provided the correct answer: momentum.

4.2 Evaluation method

For our evaluation, we used the same metrics as the official SQuAD leaderboard, which are Exact Match (EM) and F1 scores. EM measures whether the answer span matches exactly with the ground truth answer. F1 scores is computed as the harmonic mean of precision and recall, where precision is calculated as the number of correct words divided by the length of the predicted answer, and recall is calculated as the number of correct words divided by the length of the ground truth answer. For evaluation, the predicted answer is measured against 3 human answers for each question, and the highest score among the three is recorded.

4.3 Experimental details

4.3.1 Baseline Model Improvement

After adding the character embeddings to the baseline BiDAF model, we trained the model with the default parameters of the starter code. Among other things, we used 30 epochs, a learning rate of 0.001, a dropout rate of 0.2, a batch size of 64, and a hidden size of 100. To be able to compare outcomes, we did not change these parameters when we trained the BiDAF model with a different fusion function in the attention layer.

4.3.2 QANet Re-Implementation

For our implementation of the QANet model, we initially wanted to exactly reproduce the parameters described by Yu et al. in [2], with for example a hidden size of 128, 8 heads in the self-attention layer, 7 encoder blocks in the model encoder layer, with each containing 2 convolutions.

Yet, we had issues with running out of memory on the GPU. To combat this, we were forced to reduce the number of parameters of our model. We started to train one really small QANet model. Then, we tried to get closer to the original model and train bigger models, with when possible more heads in the self-attention mechanisms, more encoder blocks in the model encoder layer and an increased hidden size. We moved from an Azure NV6 to an Azure NV12 virtual machine to train our biggest model. Figure 2 contains all the details about the structure of the models we trained.

Model	QANet (1)	QANet (2)	QANet (3)
Batch Size	32	32	16
Hidden Dimensionality	64	128	128
Heads in the Self-Attention Mechanisms	3	4	8
Encoder Blocks in the Embedding Encoder Layer	1	1	1
Depthwise Separable Convolutions in each block	3	4	4
Encoder Blocks in the Model Encoder Layer	4	3	7
Depthwise Separable Convolutions in each block	3	2	2

Figure 2: Parameters of the different QANet models trained

As for the other hyperparameters, we strictly followed the instructions of the paper. We used an Adam optimizer, with $\beta_1 = 0.8$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$, a learning rate of 0.001, a dropout rate of 0.01, an exponential moving average on all trainable variables with a decay rate of 0.999. For regularization, we also used L2 weight decay with $\lambda = 3 \times 10^{-7}$ on all trainable variables. Additionally, we had dropout layers on word ($p = 0.1$) and character embeddings ($p = 0.05$). Last, in accordance with what is done in [2], we adopted the stochastic depth method (layer dropout) introduced in [9] between every layer in the encoding blocks where layer l has survival probability $p_l = 1 - \frac{l}{L}(1 - p_L)$ and where L is the last layer and $p_L = 0.9$. We trained each model for 30 epochs.

Though convolution and self-attention layers used by QANet model are better suited for GPU parallel computing than recurrent layers (left-to-right or right-to-left) in the BIDAF model, we did not observe any speed-up: it took us more than 20 hours to train each of our QANet models, the bigger models taking more time than the smaller ones. This can be explained because our self-implemented QANet models involve significantly more parameters to be learned than the BiDAF model, and also because of our potentially suboptimal implementation. As seen in Figure 1, here the context and queries are generally short in length (tens or hundreds): we could maybe see the benefit of parallel computing with longer sequences.

4.3.3 QANet Exploration

After testing different model sizes for our QANet implementation and trying to understand the effect of the number of heads and of encoder blocks, we explored some variations around the baseline architecture. We did not only see this as a way to further our understanding of how QANet works, but we also believed that it could increase the diversity of the predictions of our different models and thus improve the performance of our ensembles on the test set. We based our investigations on some of the weaknesses of the QANet model we observed and on choices made by Yu et al. in [2] for which we did not get the rationale. To better understand the exact effect of each modification we made to the QANet structure, we trained a model for each of the variations detailed below with the same hyperparameters as QANet (1).

Modified Output Layer. As detailed in the Approach part, we first tried improving the pointer mechanism of QANet's output layer by adding a linear layer between $\boldsymbol{p_{start}}$ and $\boldsymbol{p_{end}}$.

Unshared Weights. We also tested the effect of not sharing weights for the three stacks of encoder blocks in the model encoder layer. We thought of this as a way to get higher level representations of the inputs that the model would learn how to balance with lower level representations in its output layer. While it is judicious to use the same encoder for the question and the context when trying to understand how to represent natural language, it did not seem to us that logical to apply the same transformation three times after the attention layer because there is no real language behind this representation: the network could learn more from three different transformations.

Data Augmentation. Even though we did not observe any significant speed-up when training our QANet models compared with BiDAF, we still tried to investigate the effect of data augmentation on model performance. In [10], Raffel et al. indeed explain that in NLP, most improvements (unfortunately) seem to come from even more expensive models and more data. To augment data, we started to follow the idea introduced by Yu et al. in [2] of using back-translation. The idea is to use two translation models (one translation model from English to German and another translation model from German to English) to obtain paraphrases of texts; and then to append to the training data pairs of back-translated questions and their associated back-translated context. For a given question, when modifying a context, contrary to what is done by Yu et al., we only back-translated the sentences of the context that were not involved in any of the three potential answers given. The huge advantage of this approach is to avoid having to find back the answer in the produced paraphrase, which in the paper could lead to errors and mislead the model.

We used the neural machine translation model using a Transformer architecture ([5]) implemented by the fairseq team of Facebook AI available here for free. However, with this model, it took us approximately 20 seconds to back-translate each context. We would have needed more than four days of machine time to back-translate all the contexts and the questions once. Given the time constraints, we were unfortunately not able to find a way to perform this back-translation faster and thus to train a model with augmented data.

4.3.4 Model Ensembling

To increase model performance, we used all our different trained models to create an ensemble. Several approaches were investigated.

We first tried averaging the vectors p_{start} and p_{end} predicted by our different models before they are fed to the function $discretize$ that predicts the answer span when testing. We then experimented with weighted averaging, where we chose the weights manually according to the scores obtained on the development set. Another idea has been to select for a given question the answer predicted by the model that was the most certain of its prediction, e.g. for which $\boldsymbol{p_{start}}(i_{model}) \cdot \boldsymbol{p_{end}}(j_{model})$ was the highest among all models. Last, we tried majority voting directly on the CSV files: if a majority of models agreed on an answer, we picked this answer. We biased our majority voting mechanism to break ties using the predictions of the model with the highest F1-score.

4.4 Results

Model	F1	EM	AvNA
Baseline BiDAF	59.73	55.92	66.27
BiDAF + Char. Embeddings	62.92	59.60	68.93
BiDAF + Char. Embeddings + Fusion Function	**65.76**	**62.46**	**71.27**
QANet (1)	66.04	62.73	72.59
QANet (2)	68.43	64.54	74.44
QANet (3)	**68.43**	**64.91**	**74.50**
QANet (1) + Modified Output Layer	**68.22**	**64.70**	73.60
QANet (1) + Unshared Weights	66.87	63.25	**73.65**
Ensemble (Weighted Average)	70.49	67.72	75.05
Ensemble (Max Prediction)	57.55	53.96	65.01
Ensemble (Majority Voting)	**71.77**	**69.16**	**76.41**

Figure 3: Comparison of models' performances on the dev set

Figure 3 summarize the results we achieved from testing our models on the dev set. Majority voting gave us the best F1 score on the dev set. With max prediction we did not get satisfactory results at all. As of the time of writing, **with a F1 score of 68.937 and an EM score of 66.323, we occupy the 1st place of the non-PCE test leaderboard.** The differences in development and test scores could be attributed to slight differences in data distribution, and to the fact that we chose how to ensemble our models by looking at their scores on the dev set.

Overall, our QANet models performed better than the improved BiDAF models. This is consistent with what was observed on public SQuAD leaderboards. Since we have never trained two models with only one hyperparameter differing between both, we cannot from these results isolate the *ceteris paribus* effect of adding one head in the self-attention mechanism or one encoder block in the model encoder layer, and thus definitively conclude about the optimal trade-off model size/performance for QANet. Still, we manage to observe here that the batch size and hidden dimensionality play an essential role in a model's performance.

Besides, we can note that, as expected, the BiDAF model with a different fusion function got better results than the baseline one with just character embeddings: the network better learns how context and query relate to each other after the attention layer, which is useful when predicting answers.

Our modified models also improved over the QANet (1) model they are based on. For the unshared weights, one reason is that we get richer and higher level representations, which when mixed with lower level representations can account for more accurate predictions. As for the modified output layer, we believe that a reason for this improvement may have to do with the network better learning to correctly point the expected answer span.

Still, we should not forget that one major reason for these improvements is that most of them are bigger models with an extended descriptive power, involving more parameters to be learned. This may make these models more prone to overfitting in some situations. What we observed with our modified models is that when training, they tended to learn faster (20 epochs to converge for the model with unshared weights and 12 for the BiDAF with a different fusion function to compare with the 30 epochs it took to train the respective original versions of these models), but to overfit a little bit more, with the loss starting to increase after slightly less epochs (see Figure 8 in the Appendix). Our experiment with QANet (3) getting a F1 score almost equal to QANet (2) however proves that bigger models are not necessarily correlated with increased performance.

5 Analysis

5.1 Quantitative Error Analysis

Figure 4 shows the error decomposition of our top-performing model on the dev set, the ensemble model obtained with majority voting. Around 23.6% of errors are derived from Answer vs. No Answer prediction (AvNA). Interestingly, there is a significantly higher error rate of predicting unanswerable questions as answerable than predicting answerable questions as unanswerable. It is probably caused by the discrepancy in the ratio of answerable vs. unanswerable questions in the Train vs. Dev set.

Prediction / Truth	Answer	No Answer
Answer	37.2%	12.9%
No Answer	10.7%	39.2%

Figure 4: Error Decomposition on the Dev Set

When both the prediction and the truth are "Answer", 24.2% of the time, there is an error and the model fails to correctly point to the right answer. Among these errors in match, 54.9% are positional errors, where the prediction is totally off position, and 45.1% are boundary errors where the prediction shows overlap. Besides partial overlap, we also count in boundary errors the cases where the prediction is included in the answer and where the prediction includes unrelated words (till 15 in some examples). Figure 5 focuses on such errors in match.

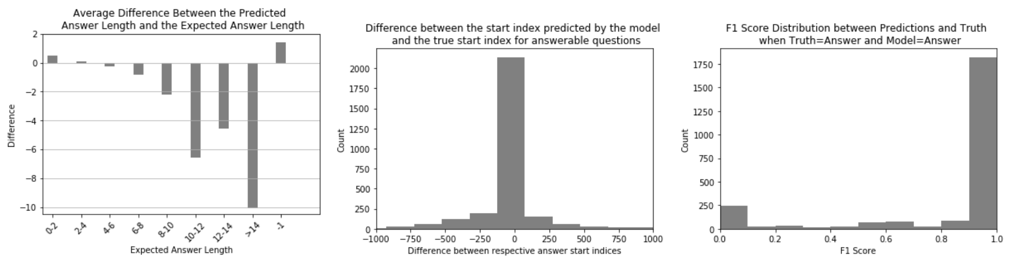


Figure 5: Key Statistics about Best Model's Predictions vs. Truth

For the few long expected expected answer lengths present in the dev set, the model tends to predict too short answers. Yet, as for the vast-majority (84%) of the answerable questions the expected answer length is comprised between 1 and 4 words, our model tends overall to predict an answer with a length close to the one of the expected answer. The second plot in the middle shows that most of the time, the model predicts the correct start pointer. This is consistent with the fact that errors in match happen only 24.2% when truth and prediction agree on "Answer".

Finally, the last plot on the right also illustrates that when it comes to errors in match, there are more positional errors than boundary error. It thus highlights that there is still future work to be done to improve even more the prediction metrics with advanced pointer mechanisms (such as using attention in the pointer generator like in [11]). Our experiment to modify the output layer was a first attempt in this direction.

To better understand how our model concretely works, it is also crucial to analyze its performance on different types of questions (Figure 6).

Type	Overall	What	Who	How	When	Where	Which	Why	Other
Count	5951	3567	627	571	455	256	214	86	65
F1	71.77	65.20	67.66	64.17	**73.14**	63.10	69.82	57.43	62.21
EM	69.16	61.80	65.69	60.17	**72.06**	58.00	65.754	48.83	58.90
AvNA	76.41	70.90	71.36	69.65	**77.60**	71.20	77.17	66.28	66.67

Figure 6: Query Types and Performance

We observe that the model performs best in answering "When", since the answers may involve numbers which are easier to retrieve. Where the model gets its worst performances is when answering "Why" questions. This was expected as these questions require some logical reasoning other than query-context matching. For each individual category, we do not observe significant performance difference when the interrogative words are placed in the start of the sentence or inside of the sentence.

Since our majority vote is biased towards top-performing QANet models, similar error patterns are found in our QANet models. When comparing the performances of our models on the different types of questions, we observe that overall, better models perform better than other models on all types of questions. Interestingly, we observe that $F1 = 64.9$ for QANet (2), $F1 = 58.5$ for BiDAF with Char. Embeddings, $F1 = 61.5$ for QANet (1) on "Why": QANet models particularly outperform on "Why" and "How" questions BiDAF models as well as QANet models with less heads in their self-attention mechanisms, indicating that self-attention is more effective in logical reasoning.

5.2 Qualitative Error Analysis

For this part, since the individual results for each question of the ensemble model are hardly interpretable in terms of being correlated with the model structure, we focus on our QANet (2) model. In Figure 9, we examine in more details some of the incorrect answers given by the model. The ground truth spans were put in bold in the table. We categorize the model's errors by a deficiency in a comprehension skill, using the skills introduced in [12].

These examples highlight the fact that our model is subject to some really common errors made by question answering deep learning models. It is not good enough yet in co-reference, inference, spatial temporal relations, as well as in basic paraphrasing.

The first example for instance shows that even though we use multi-head attention, our model is still not capturing enough of the global context, but rather relying too heavily on the local context, as the predicted answer is closer in the context to an occurrence of the word "congress" than the expected answer. More generally, in presence of multiple prolific entries, we observe that the model seems to discard lots of relevant context information, and focus on the phrases that are most similar in structure to the question. For example, although it performs especially well on "Who" and "When" questions, our model tends to fail when more than one viable answer to these questions appears.

One potential way to improve our QANet models could then be placing relatively more emphasis on self-attention (more heads) than on convolutions (less encoder blocks). Not only would it help the model better learn global dependencies, but as seen above, it could also improve the model's ability to perform logical reasoning. As for the other skills our model lacks, like co-reference, we could think of adapting ideas from state-of-the art models for each of these skills (like from [13]) to our QANet model. Using pretrained contextual embeddings, or adding other additional features to our input vectors like named entity types might also be ways to improve our models' responses to these issues.

6 Conclusion

In this project, we implemented, evaluated and analyzed several end-to-end deep learning models to perform reading comprehension problems on the SQuAD 2.0 dataset. This gave us the opportunity to gain hands-on experience on implementing deep learning architectures from scratch. We first experimented with the baseline BiDAF model, and improved its performance by modifying its embedding and attention layers. Secondly, we re-implemented the QANet model, which achieved higher performance compared to the BiDAF model. We then tested several modifications around QANet baseline structure which in their turn also increased performance.

Overall, our experiments and analysis demonstrated the power of attention mechanisms (bi-directional attention and self-attention) in deep learning models for question-answering systems. Our ensemble model indeed achieved the first place (when writing) on the non-PCE test leaderboard, with $EM = 66.32$ and $F1 = 68.94$. Future work such as performing the data augmentation as initially desired, introducing additional features to the input vectors or pre-trained contextual embedding, as well as attention in the output pointer layer, may further promote the model performance. Given that we faced memory issues on the GPUs we were working on which prevented us from training the models we wanted, we could also look for ways to improve the memory efficiency of our implementation.

7 Acknowledgements

We would like to thank the CS224N course instructors and course assistants for implementing the baseline model and providing guidance in project development, and Microsoft Azure for GPU service and support.

8 Appendix

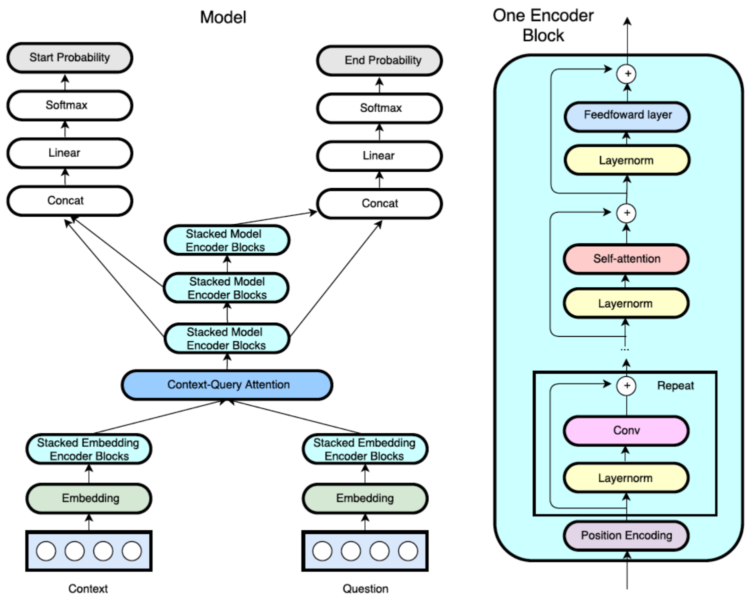
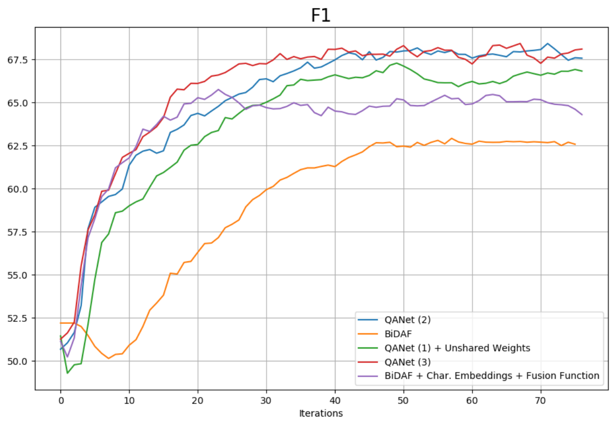


Figure 7: The overall QANet model and its encoder block from the paper by Yu et al [2]



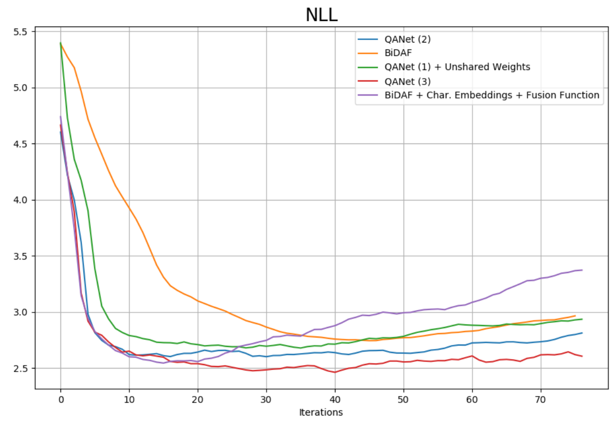


Figure 8: Evolution of the F1 score and the Loss during the training of some of our models

Shortened Context	Question	Prediction	Skill-Deficiency
The goal of the congress was to formalize a unified front in trade and negotiations with various Indians, since allegiance of the various tribes and nations was seen to be pivotal in the success in the war that was unfolding. The plan that the delegates agreed to was never ratified by the colonial legislatures nor approved of by the crown. Nevertheless, the format of the congress and many specifics of the plan **became the prototype for confederation during the War of Independence**.	What was the importance of the congress?	to formalize a unified front in trade and negotiations with various Indians	Inference
In England, the period of Norman architecture immediately succeeds that of the Anglo-Saxon and precedes the **Early Gothic**.	What architecture type came before Norman in England?		Spatio-Temporal Relations
Bethencourt took the title of King of the Canary Islands, as vassal to Henry III of Castile. In 1418, Jean's nephew Maciot de Bethencourt sold the rights to the islands to **Enrique Pérez de Guzmán**, 2nd Count de Niebla.	Who bought the rights?		Logical Reasoning
The change of control in Florida also prompted most of its Spanish Catholic population to leave. Most went to **Cuba**, including the entire governmental records from St. Augustine, although some Christianized Yamasee were resettled to the coast of Mexico.	Where did many Spanish Catholic move after British takeover in Florida?		Co-Reference

Figure 9: Examples of Errors in Predictions of Answerable Questions

References

[1] Ali Farhadi Min Joon Seo, Aniruddha Kembhavi and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. 2016.

[2] Minh-Thang Luong Rui Zhao Kai Chen Mohammad Norouzi Quoc V. Le Adams Wei Yu, David Dohan. Qanet: Combining local convolution with global self-attention for reading comprehension. 2018.

[3] Mohit Iyyer Matt Gardner Christopher Clark Kenton Lee Luke Zettlemoyer Matthew E. Peters, Mark Neumann. Deep contextualized word representations. 2017.

[4] Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. Pre-training of deep bidirectional transformers for language understanding. 2018.

[5] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need. 2017.

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[7] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[8] Yiming Yang Jaime G. Carbonell Quoc V. Le Zihang Dai, Zhilin Yang and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. 2019.

[9] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. *CoRR*, abs/1603.09382, 2016.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. 2019.

[11] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. 2018.

[12] Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. pages 806–817, July 2017.

[13] Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. *CoRR*, abs/1707.07045, 2017.