# Combining QANet and Retro-Reader Models

Stanford CS224N Default Project

**Michael Cao**
Department of Computer Science
Stanford University
caom@stanford.edu

**Isaac Cheruiyot**
Department of Computer Science
Stanford University
icykip@stanford.edu

**Atem Aguer**
Department of Computer Science
Stanford University
atemjohn@stanford.edu

## Abstract

Our task is to design a machine reading comprehension (MRC) model that can accurately solve question answering problems from the Stanford Question Answering Dataset (SQuAD). For our model, we aim to 1) implement the QANet model, which is one of the highest performing non-pretrained models, and 2) extend QANet with a verification module inspired by Zhang et al. (2020) to better identify unanswerable questions and improve performance on SQuAD 2.0. We explored variants on both the QANet architecture as well as the Retro-Reader architecture, and our best single model achieved an EM/F1 score of **66.10/62.28** on the development set and **64.422/60.659** on the test set.

## 1 Key Information to include

- Mentor: Lingjue Xie
- External Collaborators (if you have any): N/A
- Sharing project: N/A

## 2 Introduction

Automated question answering has been a task that has received significant interest in recent years, which is mostly due to the rising popularity of RNN and transformer architectures as well as the development of pre-trained contextual embeddings (PCE), all of which have led to impressive leaps in model performance on SQuAD. PCEs such as BERT and ELECTRA have led to such major improvements in performance that now models are generally split into two groups: PCE and non-PCE. Non-PCE models still rely on traditional word embeddings such as GloVe [1] and FastText [2], but as expected non-PCE models do not perform as well as PCE models. In this paper, we are looking to improve the performance on a current state-of-the-art non-PCE model, QANet, to better determine answerability of questions, as this is a major challenge introduced with the SQuAD 2.0 dataset. Because roughly half of the SQuAD2.0 dataset includes unanswerable questions, we are motivated to tailor our model to better address the challenge of MRC with unanswerable questions and focus on answerability verification by experimenting with different verification approaches and modules.

In this paper, we implement the QANet model as laid out in [3]. This model replaces the typical bidirectional encoding blocks with local convolution networks and global self-attention, and compared to the traditional RNN encoding blocks, the model is able to keep up with their F1/EM scores while both training and performing many times faster. One area where the original QANet model and other models fall short, mostly because these models were designed for the SQuAD 1.0 dataset rather

than SQuAD 2.0, is determining answerability of a question. Thus, we combine QANet with the retro-reader model from [4], which performs two stages of reading: 1) sketchy reading that skims over the passage and question and outputs an initial guess, and 2) intensive reading that utilizes the initial guess and verifies it, ultimately outputting either no answer or a final prediction.

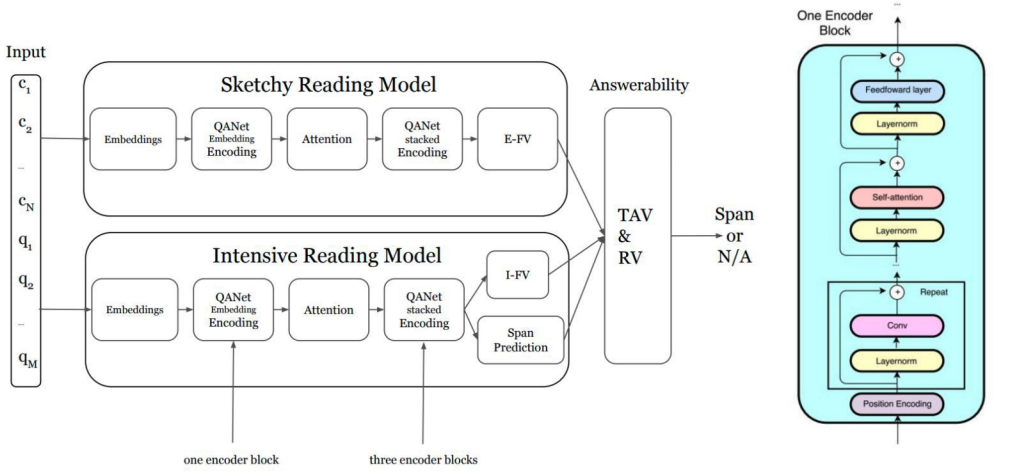
The task our model is expected to handle is question answering; our model will be given two inputs: a paragraph and a question about the paragraph, and our model will output either a span of text in the paragraph that answers the question, or <No Answer>, signifying that the question could not be answered from the information in the paragraph.

## 3 Related Work

Prior to QANet, high performing models such as BiDAF [5] generally relied on RNNs, which although effective were slow at training and evaluation leading to slow turnover in research and lack of applicability in real world situations. This was mostly due to the sequential nature of RNNs which made them difficult to parallelize. In order to solve this RNN dependency, taking inspiration and ideas from the then cutting-edge Transformer models [6], a group of researchers from Carnegie Mellon and Google Brain replaced the typical bidirectional encoding blocks with non-recurrent (and thus faster) local convolution networks and global self attention[3]. By doing so this new model was capable of keeping up with previous F1 scores while training 3 to 13 times faster and performing 4 to 9 times faster in inference. This speed improvement allowed the researchers to employ techniques such as data augmentation to increase the amount of data that is fed through the model and improve performance.

As we have begun to better understand the necessity for models to understand when there is no answer for a given question, enter verification. Verification allows a model to predict the probability of the presence of an answer within a document for a given question. One model that has emerged recently as noe of the top performers is the Retro-Reader model by Zhang et. al [4]. Developed by researchers from Shanghai Jiao Tong University, the model's usage of verification allowed it to take one of the top spots[4]. They combined the usage of internal front verification and external front verification on top of ALBERT ultimately producing an F1 score of 93.011 on SQuAD 2.0. Since this model was developed recently, the verifiers they propose have not been extensively tested on other models, which is what we aim to experiment on.

## 4 Approach




Our first step was to improve our BiDAF model's embedding layer to accommodate character embeddings, as we could also use this embedding layer for our implementation of QANet. We then implemented the QANet model following the architecture laid out in the QANet paper [3], which was made easier by our ability to reuse layers such as the embedding layer and context-query attention layer from the BiDAF baseline.

Our final model is a combination of some of the layers from our BiDAF baseline as well as several layers we implemented ourselves based on the QANet and Retro-Reader papers. Our first four layers are fairly standard layers that are common among high-performing non-PCE question answering models.

Retro-QANet is composed of three major parts: the sketchy reading module, the intensive reading module, and a rear verification module. The intuition behind the retro reader is that humans usually scan through a passage of text to make some coarse judgements about the text and make a guess about whether the question is answerable or not, and then they read more in depth to make an actual prediction for the answer. The final answerability of the question is determined by the aggregation of both of these layers.

**Embedding Layer**
This layer is similar to most input layers in other implementations. We produce an embedding for each word from concatenating its GloVe word embedding and character embedding. The character embeddings are created where their size is the input channel size of the CNN, and they and passed through a 1D convolutional layer. The outputs of the CNN are max-pooled over the entire width to obtain a fixed-size vector for each word. Similar to the BiDAF implementation, we apply a two-layer highway network [7] on top of the representation. The output has dimension of 2 times the hidden size, where the hidden size is 100. However, the outputs of the embedding layer are passed through a depthwise separable layer to resize them to 128, which is the dimension of the rest of the layers.

**Embedding Encoder Layer**
This layer is implemented with the stack of three layers: a convolution layer, a multi-head self-attention layer, and a feed-forward layer. First, we inject positional encoding into the input, since our convolutional blocks are non-recurrent and does not store positional information. The convolutional blocks utilize depthwise separable convolutions, which is more memory-efficient than standard convolutions [8], and the outputs are fed into a multi-head self-attention layer (we use PyTorch's nn.MultiHeadAttention module). Finally, we put the outputs through a feed-forward layer, which is a composition of linear and ReLU activation layers. Throughout the embedding encoder, in accordance to one of the optimizations the paper mentions, we apply stochastic layer dropout where each layer $l$ has survival probability $p_l = 1 - \frac{l(1-p_L)}{L}$. The number of convolutional layers for the embedding encoder is 2.

**Context-Query Attention Layer**
This is a standard module that is used in both the BiDAF [5] and QANet [3] models; it computes the similarities between each pair of context and query words to obtain a similarity matrix, which is then normalized and the context-to-query attention scores are computed. The similarity matrix $S$ is used to compute context-to-query and query-to-context attention:

$$A = \bar{S} \cdot Q^\top, B = \bar{S} \cdot \bar{\bar{S}} \cdot C^\top$$

Where $Q$ and $C$ are the query and context, respectively, and $\bar{S}$ and $\bar{\bar{S}}$ are the row and column-normalized matrix of $S$, respectively.

**Model Encoder Layer**
This layer uses a similar architecture as the embedding encoder layer and is comprised of a stack of 7 encoder blocks rather than 2, which is the number of convolutional layers for the embedding encoder. We run the input through our stack of 7 encoder blocks 3 times to produce 3 separate outputs.

### 4.1 Retro Reader

For our standalone QANet model, we implemented the standard QANet output layer which computes the probabilities of each position in the context being the start and end of the answer span, but our final utilizes a modified version of this output layer that also performs front verification to determine answerability.

**E-FV I-FV:**
These layers both implement the same front verification to determine whether or not the question is answerable by taking the pooled representation of the outputs of all three encoding blocks used in the model encoder layer and passing them through a fully connected layer to obtain classification logits in

accordance with Zhang et al. (2020). We then took the max difference between the answerable logits and the no-answer logits to compute our prediction. We are currently using binary cross entropy loss for both external and internal front verification:

$$h = \text{Conv1d}([M_0, M_1, M_2])$$

$$x = \text{Softmax}(h)$$

$$\hat{y_i} = max(x - x[0])$$

$$\mathbb{L}_{na} = -\frac{1}{N}\sum_1^N [y_i \log(\hat{y_i}) + (1 - y_i) \log(1 - \hat{y_i})]$$

**Span Prediction:**
This layer predicts the answer span from the three matrices from the previous step and outputs the starting and ending indices of a prediction. We adopt the strategy of Yu et al. (2018) to predict the probability of each position in the context being the start or end of an answer span. Where the start prediction is constructed of the softmax of the concatenation between the top and middle encoder layers multiplied by a learned weight, while the end is constructed similarly but with the bottom encoder instead of the middle encoder as such:

$$s = softmax(W_1[M_0; M_1]), e = softmax(W_2[M_0; M_2])$$

where $W1$ and $W2$ are two trainable variables and $M0, M1, M2$ are respectively the outputs of the three model encoders, from bottom to top. The score of a span is the maximum sum of its start position and end position probabilities. To calculate the loss for our intensive model we combine the cross entropy loss of the span prediction:

$$\mathbb{L}_{span} = -\frac{1}{N}\sum_1^N [\log(s_{y_i^s}) + \log(e_{y_i^e})]$$

with the aforementioned front verification loss:

$$\mathbb{L}_{intensive} = \mathbb{L}_{span} + \mathbb{L}_{na}$$

**Inference:**
While testing in order to make a final prediction our model utilizes an **RV & TAV Layer**; this layer inspired by Zhang et al. (2020) takes the answerablity predictions from both models and the span predictions to make a decision as to whether or not the question can be answered. To determine this we take the maximum score from the span predictions and subtract it from the no answer span prediction to get the span prediction.

$$\text{has} = max(s_k * e_l), 1 < k \leq l \leq n,$$

$$\text{null} = s_0 * e_0$$

$$\text{span prediction} = \text{null} - \text{has}$$

We then take the weighted average of the two answerability predictions from our sketchy and intensive models to produce a predicted answerability which we sum with the aforementioned span prediction. Which we finally compare to a threshold that determines whether or not the question is answerable.

$$\text{external prediction} = \beta * \hat{y_{ske}} + (1 - \beta) * \hat{y_{int}}$$

$$\text{answerable} = \lambda * \text{external prediction} + (1 - \lambda) * \text{span answerable}$$

$$\text{answerable} > \delta$$

Here we implement a novel method in which our weights($\beta, \lambda$) and threshold($\delta$) parameters are learnable. By training our inference layer we hope to improve overall performance by optimizing the emphasis placed on the answerability predictions of the sketchy model versus the intensive model as well as the boundary for which these predictions are most accurate.

# 5 Experiments

## 5.1 Data

To train, develop, and test our model, we are using the SQuAD 2.0 dataset, which contains approximately 150k questions in total. The data has been split for us into three sets: train, dev, and test. Our train set contains 129,941 examples and is identical to the official SQuAD training set, our dev set contains 6078 examples and is produced by selecting half of the official dev set, and our test set contains 5915 examples and includes the remaining examples from the official dev set plus additional hand-labeled examples. The data consists of question context pairs which when passed to a model, produces a sets of start and end pairs that determine where in the context the answer resides.

## 5.2 Evaluation method

Our evaluation metric is the F1 and EM scores the models receive. Both are percentages; EM is defined as a binary measure of whether the model's prediction exactly matches the ground truth answer, and F1 is a less strict measure that is defined as the harmonic mean of precision and recall. Another important metric that was used for evaluating our sketchy reader is AvNA, which stands for Answer vs. No Answer and measures the classification accuracy of our model when only considering whether the model outputs an answer vs. no-answer predictions.

## 5.3 Experimental details

### 5.3.1 BiDAF

For our baseline, we trained the BiDAF model with character embeddings for 30 epochs with all the default settings and hyperparameters laid out in the starter code (https://github.com/minggg/squad). Most importantly, our batch size is 64, hidden size is 100, learning rate is 0.5, and dropout probability is 0.2. Training the baseline took about 10 hours for 30 epochs.

### 5.3.2 QANet

After training the baseline, we designed and trained the QANet model by itself. For our intial implementation, we incorporated the core of what the paper outlined, but we left out some of the small optimizations. We generally followed the parameters from the QANet paper, but for our first run, instead of using the Adam optimizer we used Adadelta with the same parameters and learning rate as the BiDAF baseline. We also used a dropout rate of 0.1 for character and word embeddings as well as between layers. One issue we ran into initially was that we ran out of memory on the GPU. We moved our model to a NC6s-v3 machine, which is faster than the regular NC6 machine, but unfortunately we were not able to use a machine with more GPU memory. Thus, we reduced the size of our parameters – we used 4 attention heads instead of 8, and a batch size of 32 instead of 64.

### 5.3.3 Retro-QANet

Upon seeing success from our initial QANet implementation, we wanted to see how our intensive reader with the same parameters would perform, which would inform us at how effective I-FV is at predicting answerability. At the same time, we trained our sketchy reader with these same parameters to set us up for our full Retro-QANet model which ideally would increase the models ability to determine whether or not certain questions are answerable.

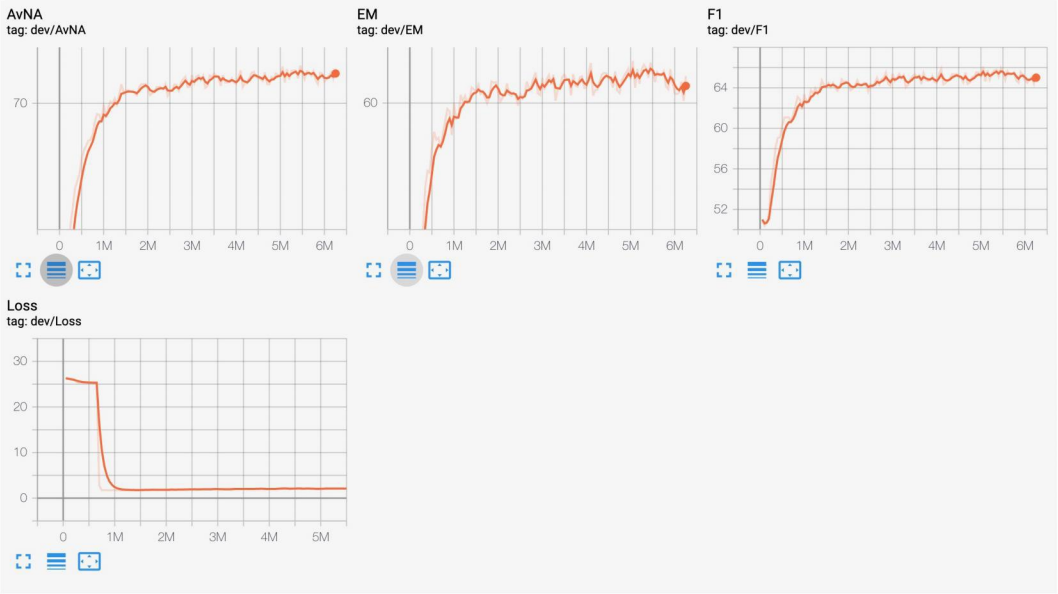
### 5.3.4 Experimentation

We performed several experiments on the intensive reader using different permutations of hyperparameters. The performance of the intensive reader tuned with different values of hyperparameters is shown in Table 1. We also made some improvements to our initial implementation of QANet; we added Xavier initializations to all the convolutional layers and we also implemented stochastic layer dropout as defined in the original paper [3]. In all the experiments, we used Adam as the optimizer and a batch size of 32. Note that the dropout probability in the table is applied only to the character embeddings layer output. The standard dropout probability used across all model output layers is 0.1 except for the character embedding layer. Unfortunately, we were unable to train our final ensemble

QANet model using the best performing Intensive reader. We made a mistake of setting the metric being maximized to the Loss instead of F1 or EM score, and therefore our intensive reader didn't save the best performing model at checkpoints. It was only later after more than 15 hours of training that we realized that this mistake had been made. And it was too late to retrain the model just in time before the deadline.

Table 1: Intensive Reader hyperparameter tuning

| Dropout Probability | No. Attention heads | F1 | EM | AvNA |
|---|---|---|---|---|
| 0.1 | 8 | **65.46** | 61.45 | **72.93** |
| 0.5 | 8 | 65.09 | **61.57** | 71.67 |
| 0.5 | 4 | 64.89 | 61.17 | 72.17 |




### 5.4 Results

Our final model achieves test set F1/EM scores of **66.10/62.28** on the development set and **64.422/60.659** on the test set placing us at — place on the IID SQuAD leaderboard. Though we see that the QANet architecture we chose to build our retro reader over performs slightly better than BiDAF, we were unable to see any speed and efficiency gains compared to the baseline. In fact, training QANet took longer than BiDAF which was likely because QANet requires many more trainable parameters. Perhaps if our input data was larger, for example if the context and/or query were much longer, we would be able to reap more of the benefits of the parallelizable nature of QANet. It is also important to note that in the original QANet paper, they performed data augmentation to the inputs which we were unable to do primarily because our implementation of QANet was not faster than BiDAF. We then decided to implement a Retro-Reader architecture around our QANet model however, we were unable to make any gains in performance.

Our initial implementation of Retro-Reader led us to quickly find that appending a retro reader provided little to no advantage to our current QANet model as we quickly found that our intensive reader predictions were already guessing no answer more often then necessary invalidating the necessity for Retro-reader. To solve this we also trained a version of QANet(Intensive Model) that did not have the ability to predict "no answer" and instead output a span prediction every time. This model, trained with the same parameters above, the only difference being a loss weight of $\frac{1}{(\text{context size})}$ was applied to the 0th position of the span prediction output.

This model reached an F1 of 38.07 however, considering we were providing a none null prediction for every question, and $50\%$ of the questions are answerable it is safe to assume we were reaching an F1 of nearly 80 on answerable questions. Thus, our goal became to reach this score by maximizing

Table 2: Model Results

| Model Name | F1 | EM | AvNA |
|---|---|---|---|
| BiDAF | 59.60 | 55.74 | 67.10 |
| BiDAF + Char Embeddings | 63.52 | 60.04 | 69.89 |
| QANet + Adadelta | 64.688 | 60.897 | 71.09 |
| Intensive + Adam | **66.10** | **62.28** | **73.11** |
| Retro-QANet | 63.72 | 59.91 | 72.32 |
| Sketchy | — | — | 63.38 |
| Naive Intensive | 38.07 | 34.69 | 47.14 |
| Retro-QANet + Naive Intensive | 57.45 | 51.03 | 61.68 |

our sketchy models prediction accuracy. Unfortunately, we have yet to reach optimal performance with our sketchy model(reaching a top AvNA of 62.38), at best we were able to improve the F1 score of our naive intensive model to 57.45, which unfortunately still lies far below our peak performers. While we would like to further experiment with our sketchy module implementation as well as with the loss weights for our intensive model, our limited time has forced us to submit our completed QANet implementation as our top performer.

# 6 Analysis

## 6.1 Error Analysis:

There are a few important factors to consider when performing error analysis for our model. One of the major factors is the different question types (What, Where, When, etc.) and how those question types are distributed across our dataset. We found that in both the training and development sets, around 60 percent of the questions were of the form "What," and the other question types, with the exception of "Why" and "Was" questions which were quite rare, were encountered in 5 to 10 percent of the examples for each category.

For our best performing model, we randomly sampled around 100 example predictions and compared them to the ground truth answers. We found that among most of the question types, the most errors were due to both false positives (predicting an answer when there is actually no answer) and false negatives (predicting no answer when there actually is an answer). For both false positives and false negatives, we encountered them 15 percent of the time, and what is interesting is that we do not see too many instances where the model predicts the wrong answer when there is an answer. We hypothesize that the reasoning behind the high incidences of false positives and false negatives is that the model is not good enough at reading comprehension, and this also possibly supports our reasoning for why our sketchy reader could not perform well at guessing answerability.

Another factor when performing error analysis is the length of answers and how far off our span predictions our in terms of comparing our predicted span length to the answer length. The vast majority of questions had answer lengths of 0-4 words, and in our sample, we found that our model was quite good at choosing answer spans that were similar lengths to the actual answer.

## 6.2 Qualitative Analysis:

While our intensive model became fairly good at predicting in answers, it generated a tendency to predict no answer more often then necessary(False Negatives) ultimately hindering its results, for example:

**Question:** Issues dealt with at Westminster are not ones who is able to deal with?
**Context:** Reserved matters are subjects that are outside the legislative competence of the Scotland Parliament. The Scottish Parliament is unable to legislate on such issues that are reserved to, and dealt with at, Westminster (and where Ministerial functions usually lie with UK Government ministers). These include abortion, broadcasting policy, civil service, common markets for UK goods and services, constitution, electricity, coal, oil, gas, nuclear energy, defence and national security,

drug policy, employment, foreign policy and relations with Europe, most aspects of transport safety and regulation, National Lottery, protection of borders, social security and stability of UK's fiscal, economic and monetary system.
**Answer:** Scottish Parliament
**Prediction:** N/A

Although a fairly straight forward, question outside of a little funky it is likely that the model would have derived an answer had, no answer not occurred as often in the testing data set. This exemplifies the issue we ran into when attempting to combine our intensive model with our sketchy model, while the sketchy model was making more accurate predictions on answerability, the intensive model had already predicted no answer in places it should not have, thus we turned to loss weighting to create a symbiotic relationship between Sketchy and Retro.

However, our model as it is, is not perfect:

**Question:** Which directive mentioned was created in 1994?
**Context:** Following the election of the UK Labour Party to government in 1997, the UK formally subscribed to the Agreement on Social Policy, which allowed it to be included with minor amendments as the Social Chapter of the 1997 Treaty of Amsterdam. The UK subsequently adopted the main legislation previously agreed under the Agreement on Social Policy, the 1994 Works Council Directive, which required workforce consultation in businesses, and the 1996 Parental Leave Directive. In the 10 years following the 1997 Treaty of Amsterdam and adoption of the Social Chapter the European Union has undertaken policy initiatives in various social policy areas, including labour and industry relations, equal opportunity, health and safety, public health, protection of children, the disabled and elderly, poverty, migrant workers, education, training and youth.

**Answer:** Works Council Directive
**Prediction:** Parental Leave Directive

From the above output, there is no explicit way of knowing what exactly in the model structure or our choices of hyperparameters may have led the model to fail to predict the correct answer. Given that one of the words in the predicted answer is contained in the correct answer, we can only hypothesize that this maybe caused by the high ratio of convolution layers which bias attention locally, to multi-headed attention layers that can capture more global information from the surrounding words. Therefore it possible that the model placed more emphasize on attending to local context than global context, and this coupled with context-question attention could have led the model to predict any answer that might have contained any of the words with in the question.

## 7 Conclusion

In this paper, we extended one of the leading non-PCE question answering model architecture to improve answerability by implementing a retro reader over the model and training both external and internal front verifiers. Overall, QANet outperformed BiDAF as expected, but we had also hoped to be able to reap some of the supposed speed and memory-efficiency gains of QANet–unfortunately, we did not see a noticeable improvement with our specific setup. Going beyond this when appending a Retro-Reader structure to our qanet model we find that performance could not be improved due to qanet's tendancy to predict no answer more often then necessary. When we attempted to correct this by having qanet prioritize correct span predictions, we saw improved results however, without a strong sketchy model to make high quality answerability predictions the model performed relatively poorly. To perform a better comparison of our baseline and QANet, we would aim to train on a GPU with more memory to remove the bottleneck to maximize the potential for QANet to outperform BiDAF due to its parallelizability. To increase the performance of Retro-Qanet we would spend more time experimenting with adjusting learning weights for our intensive reader as well as experiment with the methodologies used to extract an answerability predictions used in our front verification layer.

## 8 Acknowledgements

Our group would like to thank all the CS224N instructors and staff for designing this default final project and providing us with the preprocessed dataset, a training/testing harness, and the BiDAF model which we used to develop our baseline. We are also very grateful to be able to use Microsoft Azure to spin up VMs to train our models. Finally, we'd like to especially thank our mentor for providing us with feedback and assistance in Nooks.

## References

[1] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[3] 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. 2018.

[4] Zhuosheng Zhang, Junjie Yang, and Hai Zhao. Retrospective reader for machine reading comprehension. 2021.

[5] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[6] 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.

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

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