The goal of this project is to create a question-answering model that works well on SQuAD 2.0, which contains both difficult no-answer questions and traditional unanswerable contextual questions. BIDAF [2] used to be state-of-the-art but is an RNN-based model. It is difficult to scale to large and has trouble capturing long-term dependencies and contexts. We adapted several techniques from QANet [3] and implemented the full QANet architecture to see if transformer-based models have an architectural advantage over BIDAF. We also investigated how much model ensembling can help improve performance.

Methods

Improving BIDAF

- **Character-level Embedding:** We added character-level embedding with both d = 64 and d = 256 and concatenated it with word-level embedding.
- **Character-level CNN:** We experimented with a 2D-CNN on top of character embedding with 125 filters of kernel size (1, 3), as specified by the BIDAF paper. This output passed through MaxPooling of size (1, 16) where 16 is the maximum number of characters per word.
- **Self-Attention:** We also implemented a single self-attention layer based on the original transformer paper [2]. The self-attention output is concatenated with the BIDAF Context2Query and Query2Context attentions, shown in Figure 1.

Implementing QANet

We implemented QANet [3] from the ground up. The architecture consists of five main layers: input embedding, embedding encoder, context-query attention, model encoder, and output. We reused the BIDAF context-query attention layer due to its similarity to QANet attention.

- **3D Convolutions:** We replaced the linear projection matrices with 3D Convolutions of kernel size 1 described in the QANet paper [3].
- **Character-level CNN:** We experimented with a 2D-CNN similar to the one used in BIDAF except with 128 filters.
- **Neocortex Depth:** We implemented stochastic depth layer dropout within the embedding and model encoder layers.

![Figure 1: BIDAF with Self-Attention Layer](image)

Model Ensemble

- **Maximum Pool Probability:** For a given model, \( y_1, \ldots, y_m \in \mathbb{R} \) are the probability vectors for the start and end positions, where \( m \) is the length of the context.
  - We constructed the probability matrix for all pairs of positions \( y_{ij} \) as follows:
    \[
    P_{ij} = \text{softmax}(y_{ij})
    \]
  - Let \( P_{ij} \) for the \( i,j \) entry of matrix \( P \), then we define
    \[
    \hat{x}_i = \arg \max_{j} P_{ij}
    \]
  - Then, we pick the start and end position \( x_i \), \( x_e \) to be
    \[
    x_{i} = \arg \max_{j} P_{ij}
    \]

Analysis

- **Self-Attention with character embedding:** We observed a remarkable performance uplift over the baseline BIDAF model and its EM/F1 scores are on par with vanilla QANet.
- **QANet:** Learned much better the first few layers. Then, BIDAF continued to learn from (and overfit) the training data while QANet’s performance stabilised. This is shown in Figure 2.
- **We implemented self-attention to decrease the learning rate every 5 epochs and CycleLR to cycle between (8e-3, 1e-3, 8e-4), but they did not result in any noticeable performance improvement.**
- **BIDAF + Self-Attention took only 21h to reach EM/F1 score of 64.4/68.6. QANet took 24h for some number of epochs and more than 8 hours to reach comparable performance.**
- **Character-level CNN did not improve the performance of either BIDAF or QANet as shown in Table 1.**
- **Automatic Mixed Precision:** We observed a 10% reduction of training time for BIDAF but only a 5% reduction of training time for QANet.

![Figure 3: Train and Dev progress Log](image)

**Table 1: Performance on the SQuAD 2.0 dev set**

<table>
<thead>
<tr>
<th>Model</th>
<th>Char/Emb Dim</th>
<th>Variant</th>
<th>Exact Match</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BIDAF</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Baseline</td>
<td>59.70</td>
<td>61.25</td>
<td>68.02</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Self-Attention, Char Emb</td>
<td>61.17</td>
<td>63.48</td>
<td>70.51</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>Self-Attention, Char Emb</td>
<td>64.90</td>
<td>66.03</td>
<td>73.75</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>Self-Attention, Char Emb, Char CNN</td>
<td>63.01</td>
<td>66.07</td>
<td>71.80</td>
<td></td>
</tr>
<tr>
<td><strong>QANet</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Baseline</td>
<td>62.53</td>
<td>65.52</td>
<td>72.38</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>Baseline</td>
<td>64.02</td>
<td>66.14</td>
<td>76.79</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>Baseline</td>
<td>64.43</td>
<td>66.00</td>
<td>74.02</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>Char CNN</td>
<td>63.75</td>
<td>67.28</td>
<td>73.53</td>
<td></td>
</tr>
<tr>
<td><strong>Ensemble-5</strong></td>
<td>200</td>
<td>3 BIDAF + 2 QANet</td>
<td>69.05</td>
<td>71.78</td>
<td>76.63</td>
</tr>
<tr>
<td><strong>Ensemble-7</strong></td>
<td>200</td>
<td>4 BIDAF + 3 QANet</td>
<td>69.47</td>
<td>71.96</td>
<td>76.64</td>
</tr>
</tbody>
</table>

**Table 1:** Performance on the SQuAD 2.0 dev set

**Conclusions**

- We found that BIDAF with self-attention reached a similar performance as vanilla QANet while taking much less time to train. QANet had similar performances across all question types. BIDAF performed particularly poorly in the "why" questions, possibly a result of the inherent disadvantage of RNNs to capture long-term dependencies.
- Utilizing CNN features such as QANet significantly reduces a model’s training time.
- Character-level CNN, varying training rates and stochastic depth did not improve performance on either BIDAF or QANet. Ensemble greatly improved performance, reaching EM/F1 score of 68.47/71.96, currently top-3 on the dev set leaderboard.
- Model ensembling meant to reduce neural network’s variance, showed that combining simpler, easier to train models may produce a "bigger gain" in terms of performance given the same amount of training time.

- An EM/F1 score of ~68.45 may be the limit of the Input Embedding layer without more complex word or character embeddings. Future work should focus on whether a more complex Char/CNN layer in either encoder could improve performance.

References