Comparing Approaches to Question-Answering on SQuAD 2.0

Stanford CS224N Default Project (IID SQuAD)

Ray Iyer
Department of Computer Science
Stanford University
rri@stanford.edu

Abstract

In this project, we qualitatively and quantitatively compared two different approaches to the question-answering task on the SQuAD 2.0 dataset: 1) BiDAF: a bi-directional attention flow approach with recurrent neural networks (RNNs), and 2) QANet: a non-recurrent Transformer-based approach that relies on position encoding, convolutions, and self-attention. Our goal was to systematically evaluate claims made by the authors of the QANet paper that their proposed model is both faster to train and evaluate as well as exhibits better performance than traditional recurrence-based architectures. We were able to significantly improve the original BiDAF baseline by adding character-level embeddings to the input layer as well as introducing an MLP "fusion function" as a post-processing step to the attention flow layer, achieving **60.33 EM** and **64.19 F1** on the dev set. Furthermore, we trained multiple configurations of QANet by varying the number of stacked encoder blocks and self-attention heads to evaluate the performance/model size trade off. Our best-performing QANet model had 8 attention heads and 7 model encoder blocks with layer dropout, achieving **62.181 EM** and **65.694 F1** on the test set.

1 Key Information to include

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

2 Introduction

In task of reading comprehension, or question answering (QA), a model is given a paragraph (context) and a question about that paragraph (question) as input. The goal is to output a correct answer to the question, where the answer is a span (i.e., excerpt of text) from the context. In some cases, the question cannot be answered using the provided paragraph. This task is difficult even for humans – on the SQuAD 2.0 dataset, the official human performance against our evaluation metrics is not perfect. In the modern world, where the amount of information is continuously expanding and increasingly intractable for humans to parse through manually, it will be even more important to develop robust AI-driven solutions to precisely and accurately answer natural language questions about large bodies of unstructured text. This well-defined task represents the first step towards building such a generalized system.

In the past several years, the state-of-the-art models to tackle this task have been built on top of large pre-trained contextual embeddings (PCE), the most prominent among these being the Bidirectional Encoder Representations from Transformers (BERT) [1]. One reason for the success of PCE models is that unlike context-free embeddings like GloVE, they produce different embeddings for individual words based on the surrounding semantic context. However, in this paper we intentionally focus on non-PCE approaches since they are more amenable to end-to-end training, enable deeper customization, and are more instructive for student researchers to gain experience in building models from scratch.

In this paper, we train, evaluate, and compare two different approaches to the QA task on the Stanford Question-Answering Dataset 2.0 (SQuAD 2.0). The first is Bi-directional Attention Flow, which is provided as a baseline implementation [2]. We augment this model by adding character-level embeddings to the input layer and introducing an MLP fusion function in the attention flow layer, which both produce significant empirical performance gains. The second is QANet, which produced state-of-the-art performance on the original SQuAD dataset prior to the rise of PCE models [3]. We implement this architecture largely from scratch, using existing implementations for the self-attention, position encoding, and depthwise separable convolution sub-modules and replacing the original paper's variant of context-to-query attention with the BiDAF attention module.

In the remaining sections, we map out the landscape of existing work that inspired this project, describe our BiDAF and QANet approaches in greater technical detail, outline the setup and results for our experiments, analyze these results both quantitatively and qualitatively, and finally conclude with our key takeaways and directions for future research.

3 Related Work

Until the publication of the QANet paper, the most successful models on SQuAD 1.0 shared two key characteristics: 1) a recurrent network to process sequential input, and 2) an attention mechanism to capture long-term dependencies. Bidirectional Attention Flow (BiDAF), our baseline model, is an example of an architecture that follows this general design.

In 2017, the Transformer architecture was proposed, and it quickly drove state-of-the-art performance on a multitude of adjacent tasks, including machine translation, language generation, and language modeling [4]. A core aspect of the Transformer was its exclusive use of attention mechanisms to replace the traditional recurrent neural network in an encoder-decoder configuration.

The QANet architecture is heavily inspired by the Transformer. The authors point out that since sequential recurrent networks cannot be parallelized, they are often slow in both training and inference, which negatively effects both the research/experimentation process as well as downstream deployment (e.g., real-time applications). To solve this problem, the authors eliminate recurrent networks altogether and instead rely exclusively on feed-forward convolutions and self-attentions to separately encode the context/query, as well as traditional attention between the context and query, such that the overall computation is more easily parallelizable. The key motivation behind this model design is that convolution captures local textual structure, self-attention learns the global dependencies between word pairs, and context-query attention learns the holistic query-aware context vector to feed subsequent layers and eventually inform the output answer [3].

Due to the rapid rise of PCE methods, QANet was not officially evaluated on the new SQuAD 2.0 dataset, which contains a significant proportion of questions that cannot be answered. Thus, the goal in our paper was to validate the claims of the original paper by building a QANet model end-to-end and systematically evaluating it on the refreshed dataset, as well as comparing it to the traditional RNN-based architecture of our baseline BiDAF model.

4 Approach

4.1 Bi-directional Attention Flow (BiDAF)

The BiDAF model was provided as a complete implementation for our baseline. For additional conceptual and architectural details, please refer to the original paper [2]. In this project, we augmented this baseline model in two ways:

Character-Level Embeddings In the input layer, we add character-level embeddings. This allows the model to condition on the internal structure of words and better handle out-of-vocabulary words. To produce this per-word character-level embedding, we utilize Convolutional Neural Networks

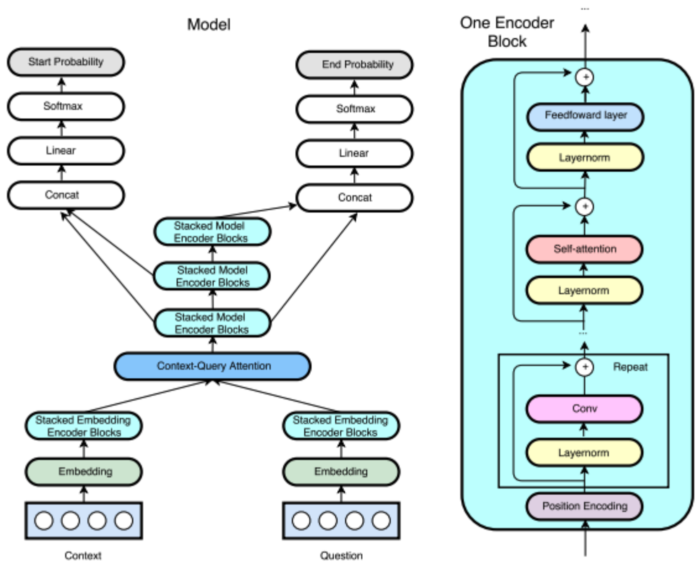

Figure 1: QANet architecture (left) with Encoder Block (right) in detail [3]

(CNNs). Specifically, our pre-trained character embeddings are used as inputs to a 1D CNN, which convolves along the length of the word. Since word lengths can differ, we then max-pool over the width of the word to obtain a fixed-size vector for each word. We then concatenate this character-level representation to our existing GLoVe embeddings for each word in the context and query before passing it to the highway network.

Attention Flow Fusion Function In the bi-directional attention flow layer, the original paper uses a simple concatenation function across the various query-to-context and context-to-query attention representations as its final output:

$$\beta(\mathbf{h}, \tilde{\mathbf{u}}, \tilde{\mathbf{h}}) = [\mathbf{h}; \tilde{\mathbf{u}}; \mathbf{h} \circ \tilde{\mathbf{u}}]$$

However, they propose that the fusion function β can be any arbitrary trainable neural network and highlight various options in Appendix B [2]. In this project, we tested applying a multi-layer perceptron (linear layer followed by a ReLU) directly after concatenation:

$$\beta(\mathbf{h}, \tilde{\mathbf{u}}, \tilde{\mathbf{h}}) = \max(\mathbf{0}, \mathbf{W}[\mathbf{h}; \tilde{\mathbf{u}}; \mathbf{h} \circ \tilde{\mathbf{u}}] + \mathbf{b})$$

4.2 QANet

The QANet model can be broken down into the following sequential modules:

Input Layer This layer is similar to the input layer of the BiDAF model. Unlike in the BiDAF model, we project the concatenated word and character embeddings for each word down to the universal hidden size of 128 prior to passing them to the Highway network, and we allow the character embeddings to be fine-tuned by the training process.

Embedding Encoder Layer This layer consists of a single Encoder Block, which is explained in detail here and also used in the model encoder layer. The weights for the query and question embedding encoder are shared. The Encoder Block consists of the following basic structure, as seen on the right side of Figure 1:

1. Position encoding
   At this step, we apply a sinusoidal function at multiple frequencies to explicitly encode the notion of position/order of single words in the context and query embeddings [4]; otherwise,

this information would be lost in the subsequent convolutional and self-attention layers, since in contrast to RNNs they do not have an inherent sense of sentence order.

2. 1D convolution layer $\times n$
The original paper proposes depthwise separable convolutions as opposed to traditional convolutions, claiming better efficiency and generalization; we replicate this in our implementation. For the embedding encoder, we use 4 convolutions with a kernel size of 7 and filter size of 128.
3. Masked multi-headed self attention layer
This is the attention mechanism originally proposed in the Transformer paper [4]. The original paper uses 8 heads, but in our experimentation we vary this and assess the results; more details below.
4. Feed-forward layer (linear -> ReLu -> linear)

Additionally, we apply layer normalization [5] and residual connections between each sub-layer operation, such that for each sub-layer f the output is

$$\texttt{out} = f(\texttt{layernorm}(x)) + x$$

Within the Encoder Block, we adapted our position encoding and self attention sub-modules from a reference PyTorch implementation [6]; all else was implemented and tested from scratch.

Context-Query Attention Layer The purpose of this layer is similar to the analogous layer in the BiDAF model: to construct a query-aware context vector for use in downstream modeling layers. While the original paper differs from BiDAF by utilizing a Dynamic Coattention Network [7] in this layer claiming a "little benefit" empirically, in our QANet implementation we use the identical implementation from our BiDAF baseline. See the BiDAF paper for a more detailed explanation of this attention mechanism [2].

Stacked Model Encoder Layer In this layer, we create a stack of 7 Encoder Blocks with the same parameters as the embedding encoder, except that 2 convolutions are applied instead of 4. The entire stack is repeatedly applied 3 times, where the weights are shared across the repetitions, which produces outputs M_1, M_2 and M_3.

Output Layer We produce two probability distributions over each position in the context that represent the predicted probability that a given position is the start or end of the true answer span, respectively. Mathematically, we return:

$$p^1 = \texttt{softmax}(W_1[M_1; M_2] + b_1), p^2 = \texttt{softmax}(W_2[M_1; M_3] + b_2)$$

where W_1, b_1, W_2, b_2 represent trainable linear layers.

5 Experiments

5.1 Data

The dataset is a version of the official SQuAD 2.0 dataset modified for CS224N.[8] Specifically, the train set is comprised of 129,941 examples, all taken from the official SQuAD 2.0 training set. The dev set is comprised of 6078 examples, which roughly corresponds to half of the official dev set randomly selected. The test set is comprised of 5915 examples, containing the remaining examples from the official dev set plus some hand-labeled examples. Each example consists of a (context, question, answer) triple, consistent with the QA task specified above. Every answerable question in the dataset has *three answers* provided, each from a different human labeler, due to the natural variance of human reading comprehension and the potential for multiple correct answers.

5.2 Evaluation method

We track three evaluation metrics to compare to my baseline BiDAF model. Exact Match (EM) and F1 scores are the official leaderboard evaluation metrics for SQuAD 2.0. EM is a binary measure of whether the model output matches the ground truth label exactly. F1 is the harmonic mean of

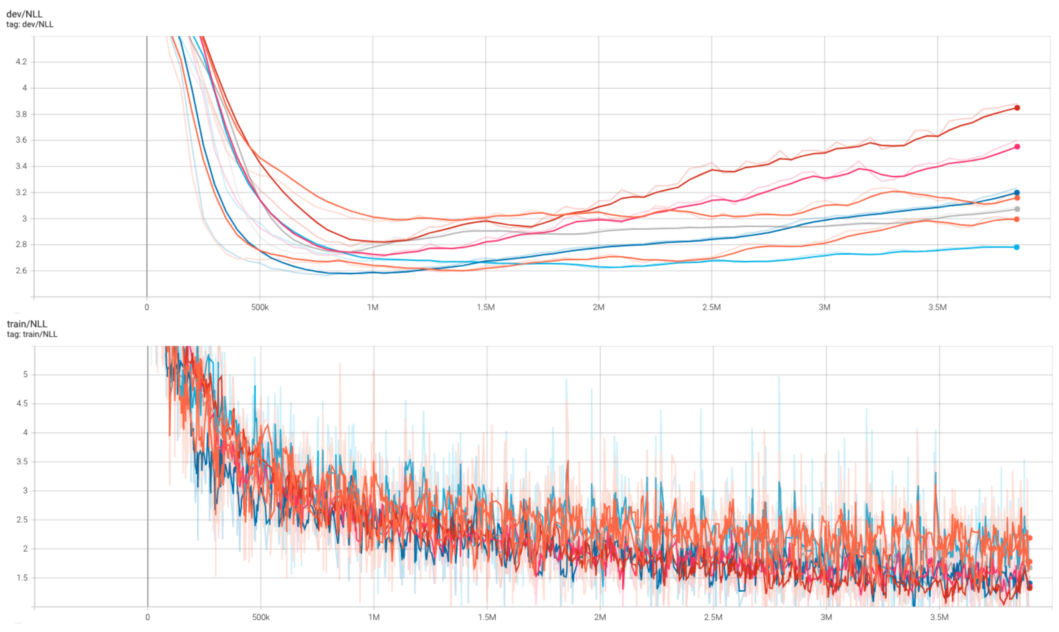

Figure 2: Train and Dev Loss: BiDAF (orange), BiDAF + char (red), BiDAF + char + fusion (pink), Small QANet (grey), Med QANet (dark blue), Orig QANet (light blue), QANet + Layer Dropout (orange)

precision and recall, which is less strict than EM. Overall, we take the maximum F1 and EM scores across the three human labeled ground truth answers, which makes our evaluation process more forgiving to the natural variation of technically correct answers. Finally, Answer vs No Answer (AvNA) is a debugging metric used to judge the binary classification accuracy of the model when predicting whether an answer exists or not.

5.3 Experimental details

All models were built in PyTorch and trained on an Azure NC6 VM with 6 vCPUs, 112 GiB RAM, supported by an NVIDIA Tesla K80 GPU.

5.3.1 BiDAF

We trained all of our BiDAF variants with the default optimizer (Adadelta), learning rate (0.5), dropout probability (0.2), and other hyperparameters established in the starter codebase.

5.3.2 QANet

We trained our QANet models with the exact specifications of the original paper: using L2 weight decay with $\lambda = 3 \times 10^{-7}$, dropout on word and character embeddings with probability 0.1 and 0.05 respectively, and general dropout between every 2 layers with probability 0.1. We also tested the effects of stochastic depth (layer dropout) [9] by training our largest QANet model both with and without it. When used, we followed the original paper by dropping a convolutional or self-attention sublayer in an Encoder Block with probability $\frac{l}{L}(0.1)$, where l is the sub-layer 1-index and L is the total number of sublayers in the stack of encoder blocks.

Since our hardware did not have the resource capacity to train with the full 32 batch size for our largest QANet models, we used the technique of *gradient accumulation* to pool gradients every two batches, which allowed our *effective* batch size to match the original paper [10].

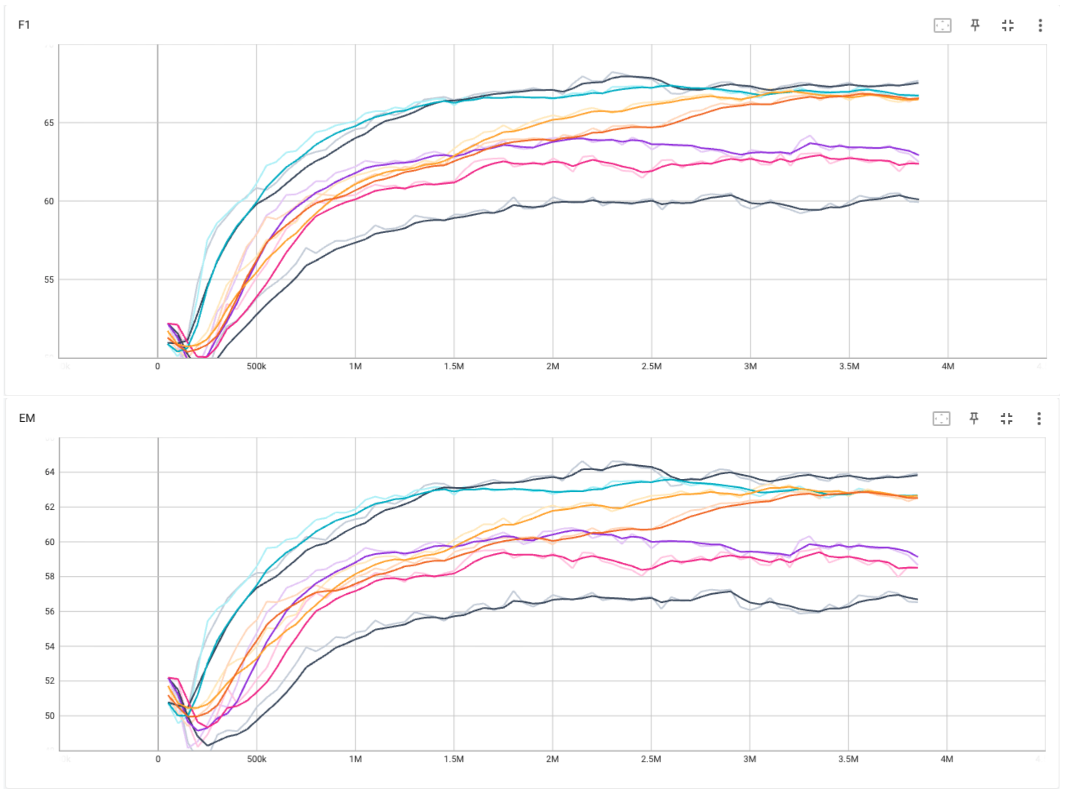

Figure 3: Dev Results: BiDAF (black), BiDAF + char (pink), BiDAF + char + fusion (purple), Small QANet (dark orange), Med QANet (turquoise), Orig QANet (light orange), QANet + Layer Dropout (black)

Model	**Batch Size**	**Train Time**	**Dev EM**	**Dev F1**	**Test EM**	**Test F1**
Baseline BiDAF	64	3h11m	57.049	60.686	*	*
BiDAF + Char Emb	64	4h49m	59.368	62.839	*	*
BiDAF + Char + Fusion Fn	**64**	**4h12m**	**60.33**	**64.19**	*	*
QANet (2 heads, 3 enc blks)	64	3h32m	62.95	67.00	*	*
QANet (4 heads, 5 enc blks)	32	6h42m	63.737	67.507	60.896	64.654
QANet (8 heads, 7 enc blks)	16	13h45m	63.27	67.13	*	*
QANet + Layer Dropout	**16**	**14h3m**	**64.54**	**68.14**	**62.181**	**65.694**

Table 1: Experimental Results

5.4 Results

As expected, QANet generally performed higher than our BiDAF baseline and all of its variants, which is consistent with the results on the official leaderboard for the SQuAD 1.1 dataset. Additionally, just as the BiDAF model EM and F1 scores drop between SQuAD 1.1 and SQuAD 2.0, we see a similar drop in empirical performance for our QANet model on SQuAD 2.0 compared to what the original paper reported on SQuAD 1.1. This is also consistent with our expectations, since the newest version of the dataset introduces questions that cannot be answered, which intuitively makes the task more difficult no matter the architecture.

It is surprising that the simple addition of an MLP fusion function to post-process the concatenation of the attention flow outputs led to such a high empirical performance gain on our BiDAF model: +0.962 EM and +1.351 F1 on the dev set. Strictly, this fusion step reduces the hidden dimensionality of the output by a factor of 4, so perhaps the performance gain arises because the model is forced to prioritize the most important information to pass downstream.

The original QANet paper claimed that the model is 3x to 13x faster in training on SQuAD 1.1. From our experimentation, we have reason to be skeptical. While our smallest QANet model which fit a batch size of 64 into our GPU memory was able to train faster than our BiDAF + Char models of equivalent batch size, it was still not by a factor of 3. Clearly, on our hardware the performance gains from parallelization are offset by the fact that this Transformer-based model is much larger than our BiDAF baseline.

Curiously, we observe that the largest QANet model, which follows the specifications of the original paper, performs slightly *worse* than the model with 4 attention heads and 5 stacked model encoder blocks. We suspect this relates to our use of stochastic depth, which will be explored more in the section below.

6 Analysis

The Model Size/Performance Tradeoff Overall, it was compelling to observe how relatively close the performance of our various QANet models were while the training speed varied so drastically due to our batch size constraints. Our results suggest that it is possible to build small models for question answering that match the performance of much larger models while still being reasonably efficient for end-to-end training, evaluation, and iteration in resource-limited research and applied settings.

It was counter-intuitive that our QANet model performance *dropped* when we increased the number of attention heads from 4 to 8 and the number of encoder blocks from 5 to 7 to match the original paper. Qualitatively, we hypothesized that the smaller model was still able to generalize well even though it does not use aggressive regularization. In contrast, as the model became larger, it was more likely to overfit on the training data. To test this hypothesis more thoroughly, we trained an additional model with stochastic depth (layer dropout) implemented as described in the original paper, aiming to conduct an ablation analysis.

Layer Dropout Ablation The results of our ablation study confirmed our hypothesis. Our large QANet model with stochastic depth clearly outperformed the medium sized QANet model. However, we note that it took almost 1.5M iterations for this performance gap to emerge, and 2M iterations for the gap to become clear, as depicted in Figure 3. Again, this confirms the intuition that deeper models require more training, but when properly regularized can deliver solid performance improvements. We also note that our QANet model with stochastic depth led to an even greater relative performance increase on the *test* set than the *dev* set: +1.285 EM, +1.04 F1 test vs +0.803 EM, +0.633 F1. Qualitatively, this makes sense given the design and motivation behind stochastic depth: we are forcing our model to learn the correct patterns even on subsets of its own architecture, such that the individual components become more robust and thus generalize better to unseen data.

7 Conclusion

In this paper, we compared two different approaches to the question-answering task on SQuAD 2.0: 1) Bi-directional Attention Flow, and 2) QANet. The former represented a traditional approach to the problem, utilizing an RNN to process sequential input and an attention mechanism to capture long-term dependencies. In contrast, the latter took inspiration from the Transformer model to do away with recurrences completely and replace them with feed-forward convolutions and self attentions such that the overall computation is more parallelizable. With the BiDAF model, we found that adding character-level embeddings and an MLP fusion function over the concatenation of attention flow outputs led to a significant empirical performance gain over the baseline. With QANet, we explored the tradeoff between model size and performance, finding that larger models require more aggressive regularization approaches like stochastic depth to combat overfitting and maintain strict performance gain; we also found that smaller models can still achieve comparatively strong performance while training far faster in resource-limited settings. Due to time and resource constraints, we were not able to vary the number of attention heads/stacked encoder blocks independently and run ablation studies on the sublayers within the encoder block, nor were we able to explore the effects of weight sharing across the encoder blocks. In the future, more systematic experimentation is required to assess and interpret the influence of individual components on the performance of a large, complex model like QANet. Given the empirical success of traditional visual recognition techniques like convolution and stochastic depth on language tasks we've seen in this project, additional research may also investigate how NLP approaches like the Transformer might drive improvements in visual recognition tasks.

References

[1] 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.

[2] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 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. *CoRR*, abs/1804.09541, 2018.

[4] 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.

[5] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[6] Bang Liu. `https://github.com/BangLiu/QANet-PyTorch`.

[7] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016.

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

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

[10] Mayukh Bhattacharyya. Gradient accumulation: Overcoming memory constraints in deep learning. `https://towardsdatascience.com/gradient-accumulation-overcoming-memory-constraints-in-deep-learning-36d411252d01`.