Convolutional Encoding in Bidirectional Attention Flow for Question Answering

Daniel R. Miller
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
danielrm@stanford.edu
Codalab User Name: danielrm

Abstract

Deep learning systems for complex natural language processing tasks like question answering are often large, cumbersome models that require excessive computational power and time. We seek to address this issue by exploring efficient and parallelizable alternatives to the more computationally expensive components of one of the top-performing question-answering architectures. In particular, we examine the use of convolutional neural networks as a replacement for the default long-short term memory units used to encode the context and question input text. The architecture we use is a streamlined version of the bidirectional attention flow model by Minjoon Seo at the Allen Institute for Artificial Intelligence.

1 Introduction

Automated reading comprehension is a powerful tool that may be used in a variety of applications to enhance data gathering and artificial intelligence models. We examine the application of modern natural language processing models to address the simplified reading comprehension problem. In this problem, a question and context pair is provided to the model, and the proper answer to the question must be extracted or generated from the context. Though simple, this is a powerful model that can be either directly used, or adapted to many different applications. For example, a virtual assistant AI such as Microsoft's Cortana, Apple's Siri, or Amazon's Echo may have such a system integrated into the back end as an information retrieval utility. Furthermore, it has been claimed that all natural language processing tasks may be seen as question answering problems [1]. In this context, the development of more powerful question answering systems may naturally lead to advances in related problems.

Deep learning systems for complex natural language processing tasks like question answering are often large, cumbersome models that require excessive computational power and time. The goal of this work is to explore alternative efficient model components that can approximate state-of-the art systems on the question answering problem. In particular, we have examined the potential replacements for the standard, slow, LSTM and GRU encoding layers. CNNs are a deep learning framework that have experienced a recent surge in popularity due to their groundbreaking success in image classification in 2012 [2]. The CNN was originally inspired by the visual cortex and its ability to focus on local regions to extract information from a larger picture. These results have motivated their subsequent use in a variety of diverse applications, ranging from audio processing for speech recognition [3] to physiological waveforms like the electroencephalogram (EEG) for clinical diagnosis and decision support [4].

Most relevant is the recent work using CNNs for natural language processing, and in particular Kim's work on sentence classification [5]. Motivated by these examples, we examine the use of

convolutional neural networks for encoding the context and query sentences used for the question answering problem. We build on a highly performing model from the Allen Institute for AI & University of Washington: Bidirectional Attention Flow [6]. Like many modern neural language processing models, Bidirectional Attention Flow, or BiDAF, uses an *attention* mechanism to focus on specific components of the input [7]. How BiDAF distinguishes itself from other common architectures is through it's titular process of encompassing not only the query-relevant context components, but also the context-relevant query components. This allows it to extract which parts of the query are actually important when asking questions about a given context. Combining these bidirectional attention mechanisms yields a more powerful model than many single-attention based methods.

While our efforts thus far have proven unsuccessful in surpassing even basic recurrent models, this project has yielded valuable insight into the relative advantages and drawbacks of various deep learning architectures. Furthermore, developing these models has provided a great deal of experience in the practical challenges involved, and motivated a very in-depth look into the various techniques that may be used to enhance under-performing models.

2 Model

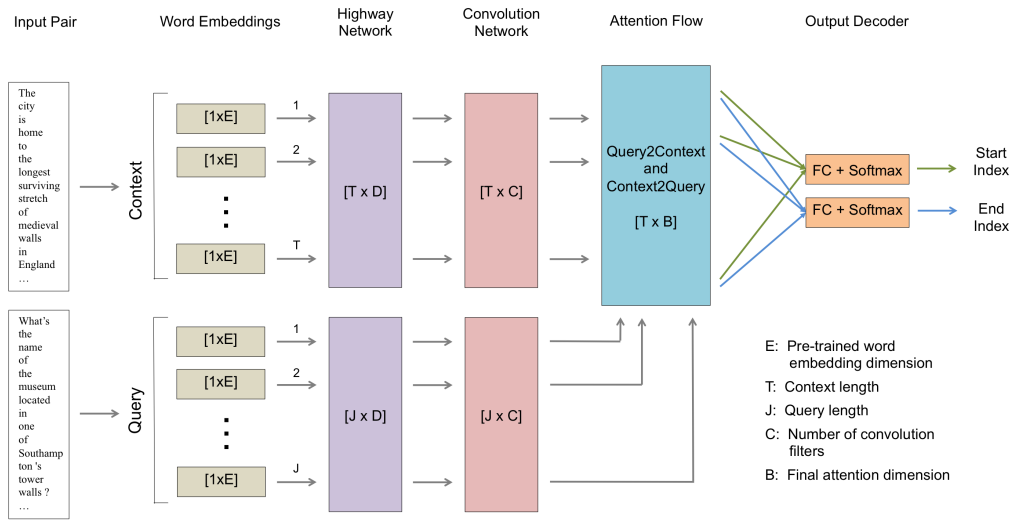

Figure 1: Bidirectional attention flow network with convolutional encoders.

Our convolutional bidirectional attention model consists of the following layers, shown in Figure 1.

1. **Word Embedding Layer:** Uses pre-trained word embeddings to map each word to a high-dimensional vector in a co-occurrence-based semantic space.
2. **Highway Network:** Allows the model to train a problem-specific set of word embeddings.
3. **Convolutional Encoding Layer:** Generates a sentence summary encoding for both the context and query that captures word interactions.
4. **Attention Flow Layer:** Combines the encodings for the context and query to capture a bidirectional attention flow.
5. **Output Layer:** Fully-connected layer with distinct softmax outputs for the span start and end position classifiers.

Word Embedding Layer: The inputs to this layer are a set of word indices for the context: $\tilde{x}_1, ..., \tilde{x}_T$, and for the query: $\tilde{q}_1, ..., \tilde{q}_J$. These indices are mapped into the co-occurrence based semantic GloVe word embedding space [8]. We used the pre-trained $E = 300$-dimensional word embeddings generated from the 840-billion token Common Crawl corpus [9]. This yields a context

representation $\tilde{X} \in \mathbb{R}^{T \times E}$, where T is the context length (number of words), and $\tilde{Q} \in \mathbb{R}^{J \times E}$, where J is the query length. In our training data, the maximum context length was 766, and the maximum question length was 76. Our pre-processing step clipped and padded all inputs to 750 and 75 respectively, to allow for mini-batching in our optimization.

Highway Network: The pre-trained word embeddings are fed into a highway network [10]. This allows for us to train a problem-specific adaption of the word embeddings, yielding our final representations $X \in \mathbb{R}^{T \times E}$, and $Q \in \mathbb{R}^{J \times E}$.

Convolutional Encoding Layer: The representations X and Q are then encoded using a one dimensional convolutional neural network with C filters, each of width w. This yields an encoded context $H \in \mathbb{R}^{T \times C}$, and an encoded query $U \in \mathbb{R}^{J \times C}$. We use a single stride and zero-padding on the boundaries to retain the "time"/sentence length. This is motivated by the recurrent neural network layers that yield an encoded representation at each time step. Maintaining this structure allows us to use a drag-and-drop approach to examine CNNs as alternative encoding structures for a variety of different developed models. Our work is motivated by previous successes in using CNNs for sentence classification [5]. Future work may include the design of more complex convolutional encoding layers. A first step would be to incorporate multiple filter widths. Beyond this, we could consider deep convolutional networks, and dimensional reduction via pooling followed by upsampling / transpose convolution.

Attention Flow Layer: As with the inspiration BiDAF model, we here compute attention in two directions, from query to context, and context to query. As in a normal attention model, this allows us to take into account how relevant each word in the context is with respect to the specific query, allowing for adaptation to variety of queries about a fixed context. Beyond this, we can also take into account the reversed attention, and build a representation of the query that is specific to the current context. This allows for more flexibility and shared information if the same question were to be asked about more than one context.

To begin, we compute the similarity matrix $S \in \mathbb{R}^{T \times J}$. $S_{t,j}$ represents the similarity between the t-th context word, and the j-th query word by:

$$S_{t,j} = w^T \left[H(t,:); U(j,:); H(t,:) \circ U(j,:) \right]. \tag{1}$$

with trainable weight $w \in \mathbb{R}^{3C}$.

We then compute the context-to-query attention $\tilde{U}$ by:

$$a = softmax(S) \in \mathbb{R}^{T \times J} \text{ over the columns, and}$$
$$\tilde{U} = a \cdot U$$

The query-to-context attention is constucted in a similar manner, but expanding to align the dimensions:

$$b = softmax(\max_{(col)}(S)) \in \mathbb{R}^T$$
$$\tilde{h} = b \cdot H \in \mathbb{R}^C, \text{ and tiling by row to:}$$
$$\tilde{H} = tile(\tilde{h}) \in \mathbb{R}^{T \times C}$$

These attentions are then concatenated into a single bidirectional attention representation:

$$G = \left[H; \tilde{U}; H \circ \tilde{U}; H \circ \tilde{H} \right] \in \mathbb{R}^{Tx4C} \tag{2}$$

Output Layer: The combined attention is then used in two separate fully connected layers, each with a softmax classifier to generate the span start and end index probability distributions $p_1, p_2 \in \mathbb{R}^T$.

Training: Our loss function for each context / query pair is the sum of the cross-entropy error for the start and end predicted softmax probabilities:

$$L = CE(p_1, y_1) + CE(p_2, y_2) = -\log p_{1,start} - \log p_{2,stop} \tag{3}$$

This loss is averaged across all samples to form the total loss.

Evaluation: For the final answer generation and evaluations, we solve the constrained optimization problem:

$$\begin{aligned} \text{(start, stop)} = \text{maximize } & p_{1,k} \cdot p_{2,l} \\ \text{such that } & k \leq l \leq T \end{aligned}$$

3 Previous Work

This model is heavily inspired by the original BiDAF model by Seo et. al. that holds third place in the open SQuAD leaderboard at the time of this paper [6, 11]. We also draw inspiration from Kim's paper on sentence classification using CNNs [5].

4 Experiments

All experiments were performed using the Python package TensorFlow for deep learning, and SciPy's Numpy for data loading and manipulation [12, 13, 14].

Optimizer: We used Adam optimization to train our model [15]. On top of this, we implemented gradient clipping to prevent exploding gradient issues, though this was more concerning for the recurrent network baselines we considered than for the CNN-based model. We swept the maximum gradient norm clipping value from 0.01 to 10, with the best results achieved at 1.

Learning Rate Annealing: We swept the learning rate from 0.00001 to 0.5. Our best results used an initial learning rate of 0.005, and annealed with a step decay of 0.9 per epoch. This allowed our model to train more quickly in early time steps, and reduced volatility in later stages of training.

Dropout Regularization: One of the primary issues with this model was a propensity to overfit the training data, without transferring comparable performance to the validation or test sets. In many deep learning models, dropout forms a useful framework to train redundancies and model robustness, and help reduce overfitting [16]. We therefore experimented with a wide array of dropout rates and locations. Dropout was implemented on the convolutional encoders H and U, and on the combined coattention representation G. We tested with extremely aggressive dropout rates up to 0.95, which improved the relative discrepancy between training and validation results, but at the cost of overall performance. Our best results were achieved with a dropout rate of 0.2.

L_2 Regularization: We also experimented with L_2 regularization to train a modified loss objective over the weighted sum of the original loss with the total L_2 norm of all parameters as follows:

$$L = CE(p_1, y_1) + CE(p_2, y_2) + \lambda \cdot \sum_{W \in \theta} \|W\|_2 \tag{4}$$

where θ represents our parameter space. This conferred only minor improvements. The best results were achieved with $\lambda = 0.0001$. However, this hyperparameter is highly interactive with the total parameters used, which varies by the number of hidden states and the filter widths. Because of the relative difficulty with tuning, we considered this hyperparameter last, and did not test against the full range of other hyperparameter values.

Batch Normalization: We hypothesized that much of the overfitting was caused by a covariate shift between the training and validation sets. This idea is supported by the differences in vocabulary between the sets. To address this issue, we implemented batch normalization to all model parameters [17]. Our best results were achieved with a decay rate of 0.95.

Remaining Parameters: The relative speed of our model allowed for extensive exploration of the hyperparameter space. However, as the CNN-based model resulted in relatively poor overall performance, these results do not vary significantly. Therefore, the full ablation results are less relevant and are not included here. The best results were achieved with a batch size of 100, a state size of 128, and a single convolutional filter width of 8.

5 Results

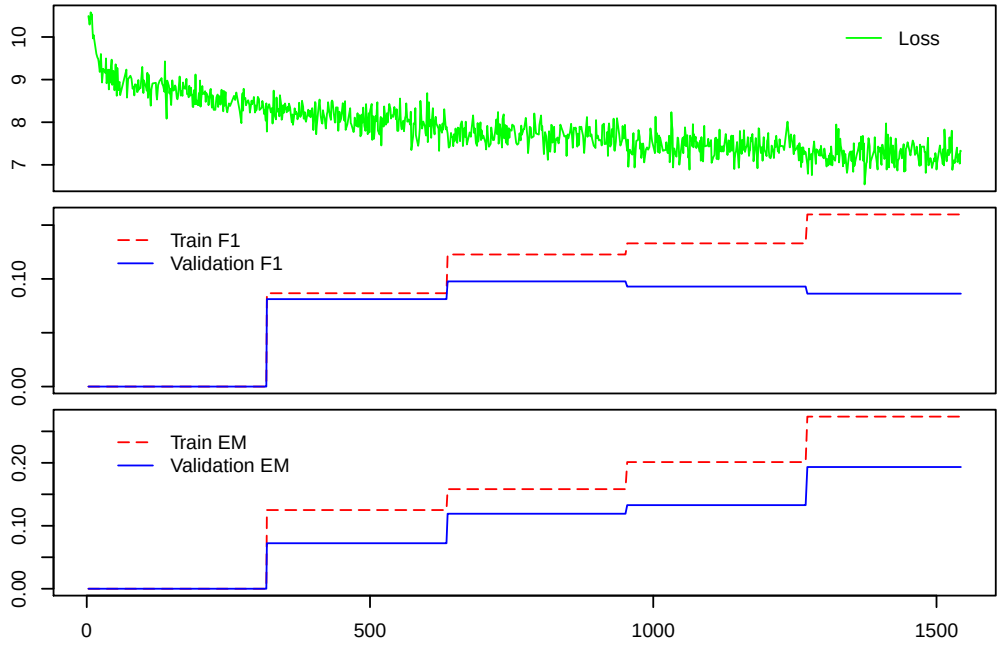

Figure 2: Training results for the bidirectional attention flow model with convolutional encoders. 5 training epochs are shown, with the F1 and EM statistics evaluated on a sample of size 400 after each epoch.

The full training and validation results of our model with the parameters specified above are shown in Figure 2. Even after 3 epochs, the over-fitting behavior begins to emerge when viewing the validation $F1$ dropoff.

We can view the much longer-term behavior over 80 epochs in Figure 3. Although the training $F1$ continues to slowly increase, the validation metric quickly flattens out.

This model achieved a final dev score of $F1 = 13.375$, and $EM = 1.22$. The results on the test leaderboard were $F1 = 13.756$, and $EM = 1.584$.

This model clearly underperformed in comparison to even baseline recurrent neural network-based models.

6 Conclusions

Reflection: The focus of this work was to explore efficient alternatives to the state-of-the art recurrent neural network based models that are most common in the reading comprehension task. In particular, we hoped that the high degree of parallelization inherent in convolutional models could lead to more efficient algorithms with much faster training times, particularly when implemented on a GPU. While this did turn out to be the case, the convolutional structures were ultimately were unsuccessful in capturing a useful encoded sentence representation. The extreme speedup allowed for much larger models to be trained for many epochs in a relatively short time frame, but this approach led to a direct memorization of the training set, with no appreciable transfer of capability to the development sets. Most of our work sought to address this issue by pursuing aggressive regularization techniques, and taking advantage of the model efficiency to quickly search the hyperparameter space.

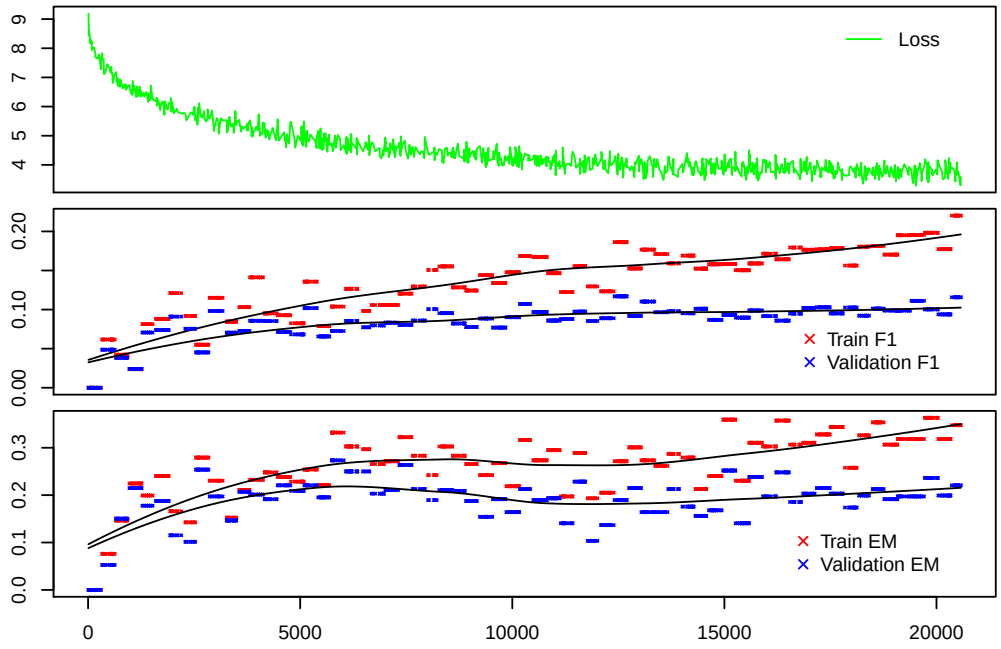

Figure 3: Long-term training results. Results from 80 epochs shown, with trendlines.

Unfortunately, our this model's performance ultimately has proven unsuccessful in the SQuAD question-answering problem. However, the experience gained in building a training framework, implementing a range of published models, and debugging a deep learning network has been one of the most practically useful parts of this class. Our model's poor performance motivated researching and implementing a wide range of regularization and batch normalization techniques, and provided a window into the practical challenges of deep learning.

Future Work: In the future, we would like to consider more complex convolutional encoding layers, and other alternatives to the standard recurrent models. In particular, Quasi-RNNs may be an effective compromise between the convolutional network's efficiency, and the recurrent network's representational power [18]. Additionally, a deeper multilayer convolutional encoding structure with variable filter widths may be more successful in capturing a useful context and query representation.

We would also like to consider further speed-based optimizations. In particular, length-based input bucketing of the context and question pairs may allow for a much more efficient implementation, regardless of the model used.

Acknowledgments

Many thanks to Richard Socher, Christopher Manning, and all Winter 2017 teaching staff for CS 224n: Natural Language Processing with Deep Learning at Stanford University. This course provided an extremely educational and challenging quarter.

References

[1] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," *CoRR, abs/1506.07285*, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[3] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 4277–4280, IEEE, 2012.

[4] P. W. Mirowski, Y. LeCun, D. Madhavan, and R. Kuzniecky, "Comparing svm and convolutional networks for epileptic seizure prediction from intracranial eeg," in *Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on*, pp. 244–249, IEEE, 2008.

[5] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[6] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603*, 2016.

[7] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[8] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[9] "Common Crawl." https://commoncrawl.org/.

[10] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[11] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[12] "Python language reference, version 2.7." http://www.python.org. Python Software Foundation.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[14] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed 2017-03-21].

[15] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[18] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-recurrent neural networks," *arXiv preprint arXiv:1611.01576*, 2016.