Relevancy-Scaled Deep Co-Attentive Networks for Question Answering

Nadav Hollander
Stanford University
nadavh@stanford.edu

Varun Gupta
Stanford University
vgupta16@stanford.edu

Orry Despo
Stanford University
odespo@stanford.edu

Test-set CodaLab Metrics found under user-name nadavh

1 Introduction

Question answering is a task in artificial intelligence oriented towards building machines that can understand a given passage and then answer different types of questions about the passage. Effective question answering poses practical implications for automating important knowledge discovery and information retrieval systems; for instance, improved question answering systems could augment the ability of physicians to practice evidence-based clinical medicine[1]. Initial attempts at standardizing the evaluation of QA systems have produced benchmark datasets that lack either the volume necessary to train complex deep learning models – a flaw of small yet high-quality human-annotated datasets (CITE) – or the diversity of question-answer types that require multiple modes of logical reasoning - flaw of automatically generated datasets (CITE). Rajurkar et al.[2] used crowdsourcing to generate a dataset significantly larger than previous hand-annotated question-answer datasets, which presents a salient opportunity for furthering research in the field of question answering. In this paper, we attempt to construct a performant model for question answering using the SQuAD dataset.

2 Related Work

2.1 Statistical Question Answering

Similar to other early approaches to most tasks in artificial intelligence, early approaches to question-answering explored the possibility of using logic and theorem-proving to build effective models[3]. In a similar vein, some approaches aimed to discover certain probabilistic inference rules for question-answering[4]. Later advances in web-scale infrastructure combined with emergent search engines such as Google and AskJeeves yielded statistical models for identifying documents most relevant to particular questions in information retrieval research[5]. While these models placed a greater emphasis on identifying relevant passages that could answer a particular question rather than identifying a specific answer given a question and a relevant passage, elements of these models have proved useful for neural question answering models. The use of text summarization methods and linguistic filters to scan for passages most relevant to particular questions[6] motivates the use of similar relevancy scaling or filtering in the model presented by this paper. An emphasis on representing the related passage with some deep and structured representation against which to execute a separate representation of the question also emerged out of statistical question answering. On the heels of such statistical research, virtually all neural question answering systems attempt to represent passages and questions using shared but possibly variable deep representations.

2.2 Neural Question Answering

Rather than mapping questions to formal queries, using word co-occurrences in order to match portions of a passage to a question, or taking advantage of hand-crafted features in multicomponent QA systems, end-to-end neural question answering systems rely on recurrent networks to produce well-formed hidden representations of input questions and passages and on novel attention mechanisms to identify answer spans in the passages.

Wang et al.[7] used relevancy scaling in order to down-weight the importance of passage words that did not have much in common with any of the words in a question; the authors also used bidirectional-LSTMs and concatenated the forward and backward hidden states at each time step at several points through their model. The authors use a decoder that separately predicts a distribution over the passage for start indexes and a distribution over the passage for end indexes. We similarly apply relevancy scaling and concatenated bi-LSTM outputs in our model.

Xiong et al.[8] used a single bidirectional-LSTM to encode the initial temporal relationships of the passage and the question respectively in order to share representation power and then use a fully-connected network and nonlinearity to optionally enable variation in the question encoding space. They also extensively describe a novel coattention mechanism that simultaneously attends to both the question and the passage and fuses those contexts together into one representation.

Inspired by the above works, we propose a neural QA model with two subsystems: first, an attentive encoder that uses relevancy filtering and shared representation with optional variation to produce temporally-aware question-passage summaries, and second, a decoder that feeds in the predicted distribution of start points as another input into the prediction of the end point distribution.

3 Dataset

The Stanford Question Answering Dataset (SQuAD) is a crowd-sourced reading comprehension dataset comprised of over 100,000+ question-answer pairs corresponding to 500+ passages of context data. For any given passage and question, the corresponding answer is represented as a substring of the passage. SQuAD is significantly larger than comparable datasets in the realm of contextual question-answering and thus represents a respectable yardstick to measure performance against.

Question: Why was Tesla returned to Gospic?
Passage: On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. During that year, Tesla taught a large class of students in his old school, Higher Real Gymnasium, in Gospic.
Answer: not having a residence permit

Figure 1: Example Question-Passage-Answer Pair from SQuAD

4 Approach

4.1 Overview

We attempt to combine elements from existing state-of-the-art models[7,8] with a unique decoder mechanism outlined below. Moreover, we attempt to augment our model's performance by exploring alternative mechanisms for computing loss and interpreting outputted probability distributions. Our model leverages pre-trained GloVe[9] word embeddings to provide us with numerical vector representations of words in the given passages and questions.

4.1.1 Relevancy Scaling

Our model begins by leveraging relevancy scaling[7] in order to filter the passage for sections more relevant to the question at hand. Let $X^Q = (q_1, q_2, , q_n)$ be the sequence of word vectors corresponding to words in the question, and $X^P = (p_1, p_2, , p_m)$ be the sequence of word vectors corresponding to words in the passage. We compute cosine similarity matrix R and scale vectors $p_i \in X^P$ such that

$$R_{ij} = \frac{p_i \cdot q_j}{||p_i|| ||q_j||}$$

$$X_i^P = p_i * max(R_{i1}, R_{i2}, ..., R_{in})$$

We run the unscaled question embeddings and scaled passage embeddings through two separate bidirectional Long Short Term Memory Networks (LSTMs)[10] in order to produce representations P and Q.

4.2 Encoding Layer

The representations are fed into a co-attention mechanism based on the aformentioned model proposed by Xiong et al.[8] . We compute a raw co-attention matrix $L = P^T Q$, and compute attention weights as follows

$$A^Q = softmax(L)$$

$$A^D = softmax(L^T)$$

The attention weights are used to summarize the document in terms of each word in the question with $C^Q = PA^Q$. Similarly, we compute summaries in light of each word in the passage with $C^D = [Q; C^Q]A^D$, where $[a; b]$ denotes concatenation. The resultant co-attentive representation $[C^D, P]$ is fed into yet another bi-directional LSTM, outputting our final knowledge representation for our encoder.

4.3 Decoder Layer

Given a knowledge representation $U = [u_1, u_2, ..., u_m]$, we first feed it into yet another bi-directional LSTM, the outputs of which we define as $A = [a_1, a_2, ..., a_m]$. We then apply a fully-connected nueral network layer to A in order to compute

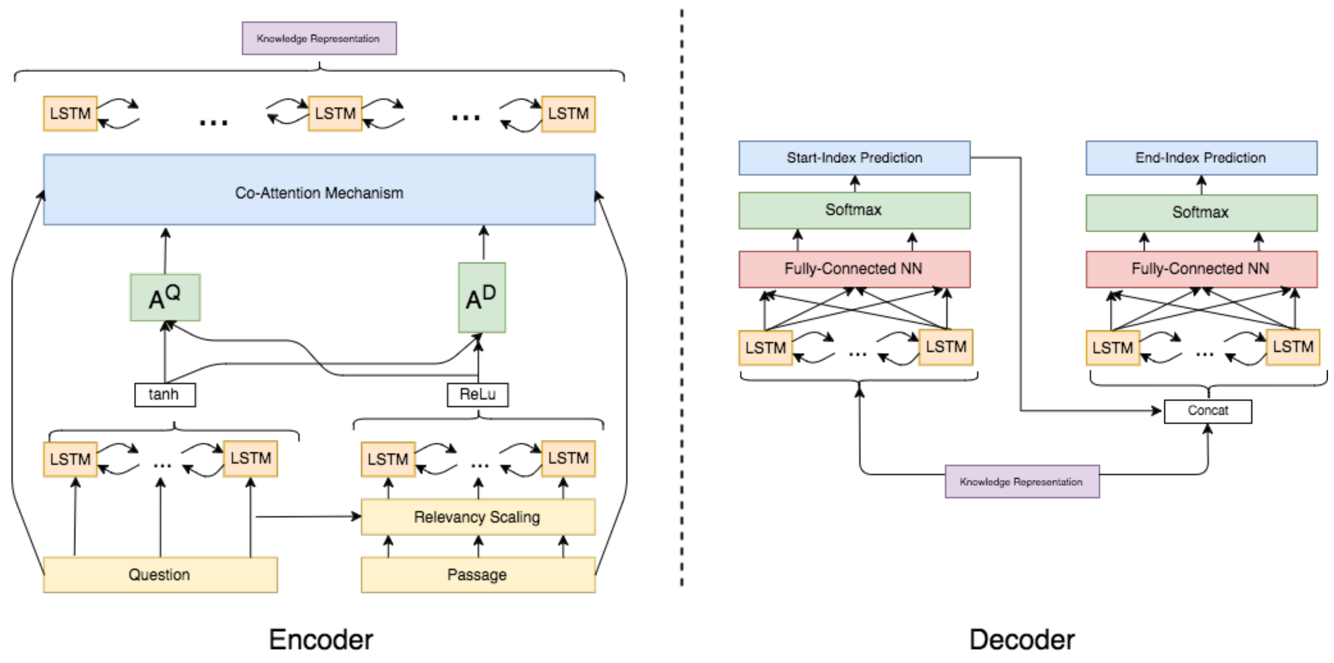

Figure 2: Relevancy Scaled Co-Attentive Model Architecture

logits to be fed into a softmax function, which allows us to compute a probability distribution $A' = softmax(W_A A + b_a)$ representing the likelihood of indices being the answer's start index. We then concatenate A' to our original knowledge representation U and feed $[U; A']$ into another bidirectional LSTM, the outputs of which we define as $B = [b_1, b_2, ...b_m]$. Once again, we apply a fully-connected nueral network layer to B, giving us logits that are fed into the softmax function in order to compute the probability distribution $B' = softmax(W_B B + b_B)$ representing the likelihood of indices being the answer's end index. In our baseline implementation, we compute our start and end index predictions s_t, e_t as

$$s_t = argmax(A') \quad e_t = argmax(B')$$

End index predictions enforced to follow start index predictions.

4.4 Gaussian Cross-Entropy Loss (GCE)

Given the ordinal nature of the indices, cross-entropy loss computed using one-hot vectors fails to capture the preferrability of incorrect index predictions that are closer to the true index over those that are further away. We attempt to extend our model by computing a cross entropy loss using Gaussian functions centered at the true indices in lieu of one-hot vectors, as seen as follows (where $g(y, \sigma)$ is the Gaussian filter centered at y with standard deviation σ):

$$GCE(y, y') = -\sum_i g(y, \sigma) log(y')$$

4.4.1 Dynamic Programming Answer Mechanism

Another augmentation we attempt on our model is a dynamic programming mechanism to replace the index selection process described above. Instead of simply choosing the start index as the $argmax(A')$, we solve the following optimization:

$$\begin{aligned} \underset{s_t, e_t}{\text{argmax}} \quad & A'(s_t) + B'(e_t) \\ \text{subject to} \quad & e_t \geq s_t \end{aligned}$$

4.5 Training Implementation

During training time, we use a max passage sequence length of 324 and a max question sequence length of 23, both of which are the 99th percentile of their respective empirical distributions in the SQuAD dataset. We pad those passages and questions with length less than their respective maximums and discard those passages and questions with length greater than their respective maximums. During validation and test time, we define the maximums to be the respective maximums in the entire dataset, and thus pad sequences appropriately. As determined by Xiong et al.[8], we also found that that training the word vector embeddings led to overfitting and therefore reduced the ability of the network to generalize to unseen data. We therefore report results with constant word embeddings. We use a hidden size of 100 throughout our recurrent units and linear layers. We initialize weight matrices using Xavier initialization and bias vectors using constant initialization at zero. We use dropout with the recommended keep probability of 0.5 to regularize the model network during training[11]. We optimize the model using ADAM and use an exponentially decaying rate to preserve training momentum and ensure a model fit.[12]

5 Results and Analysis

5.1 Experimental Path

As we later demonstrate, our final model achieves subpar performance. However, it represents a cumulation of a long series of insightful exploration and experiments, which we describe below.

5.1.1 Prediction Modeling

We experimented with a variety of methods to model how to predict the answer over a passage. The first representation was simply modeling a distribution over each word of whether it was in the answer or not. Empirically, this lead to ~ 0 F1/EM. Qualitatively, we found each word to be substantially more likely to not be in the answer, giving us unusually short predictions. Unless we set our threshold probability extremely low, we'd predict nothing for each sentence. To combat this, we linearly scaled our probabilities down and implemented Kadane's algorithm to find the maximum contiguous subsequence of answer probabilities. Unfortunately, we were still left with extremely short predictions.

Given this, we decided to predict the start and end indices directly with the hope that it would be easier to predict two items per passage rather than multiple. Particularly, we switched to modeling two separate probability distributions for each index that denoted whether a certain index was the start index (or not) and whether the index was the end index (or not). We did this by simply taking the knowledge representation and feeding it into two seperate fully connected layers (one for the start and one for the end). After, the model softmaxed over each word and then took the argmax out of all of the start index probabilities as the start index. This lead to an improvement over our previous implementation. However, qualitatively, our predicted answers were much longer than the ground truth answers. We attribute this to doing a softmax on a per-word basis instead of doing a softmax over the whole passage, which we subsequently changed to. This lead us to achieve predicted answer lengths much more inline with the ground truth.

Finally, we noticed instances where our model was not confident on any of the words to be start indices i.e. it predicted them all to have similar probabilities. However, it clearly predicted a word to be the end index. In this case, it would sometimes choose a start index that would be past the end index, resulting in a completely wrong prediction. Thus, to help us not land in these local minimas, we decided to choose the start index and end index simultaneously. In other words, we chose the pair (s_t, e_t) such that the sum of the start index probability and end index probability was maximized.

5.1.2 Decoder Architecture

Given our architecture choices, the job of the decoder was to take the knowledge representation (i.e. the integrated information of the passage and question) and identify likely start and end indices. The first model was to simply use two separate fully connected layers (along with softmax) to predict the start and end index distributions for each passage. However, we noticed we had trouble predicting answers that required logic to span multiple lines. To enable more complex decoding, we experimented with replacing each of the fully connected layers with two fully connected layers (with Relu and dropout in between). This lead to small empirical improvements, but the fundamental problem still persisted.

Thus, we experimented with passing the knowledge representation through a Bi-LSTM, concatenating the forward and backward hidden states, then passing each word into a fully connected layer to get the start index distribution. We did the same for the end-index. This produced much stronger empirical results.

Finally, given that we as humans usually choose the beginning of an answer before the end, we experimented with conditioning on the start-index distributions while performing the end-index Bi-LSTM operation, which also proved fruitful.

5.1.3 Quadratic Representation

One of the features of our coattention is calculating an affinity matrix, which functions as the raw co-attention matrix. Specifically, it takes the form $L = P^T Q$, where P and Q represent passage and word encodings. However, this creates L uniformly across P and Q. To allow for this relationship to be learned, we instead simulated a quadratic form for this by inserting a fully connected layer i.e. $L = (P^T W + b)Q$. Unfortunately, this produced little empirical improvement and consistently led to drastic over-fitting.

Model	Dev F1 / EM	Test F1 / EM*
Baseline Model (Hidden Size 200)	39.08 / 23.5	-
Baseline Model (Hidden Size 100)	39.67 / 29.0	-
LSTM Decoder (Hidden Size 100)	43.07 / 31.0	-
LSTM Decoder (Glove Dimensionality 200)	44.12 / 30.25	-
LSTM Decoder w/ DP Mechanism	**44.33 / 30.30**	**43.15 / 30.87**
LSTM Decoder w/ GCE Loss	43.88 / 31.05	-

Figure 3: Experimental results using varied hyper-parameterizations and model extensions

5.2 Evaluation-Phase Results

Given the aforementioned experiments, we converged on a model described in section 4. As can be seen in Figure 3, the Relevancy-Scaled Co-Attentive Model with the bi-directional LSTM decoder described in section 5.1.2 managed to achieve a marked improvement over our baseline bi-directional LSTM encoder / decoder model. We found that, empirically, the model augmented with the aforementioned Dynamic Programming mechanism proved most performant, while the experimental Gaussian Cross Entropy Loss proved relatively unfruitful in its results.

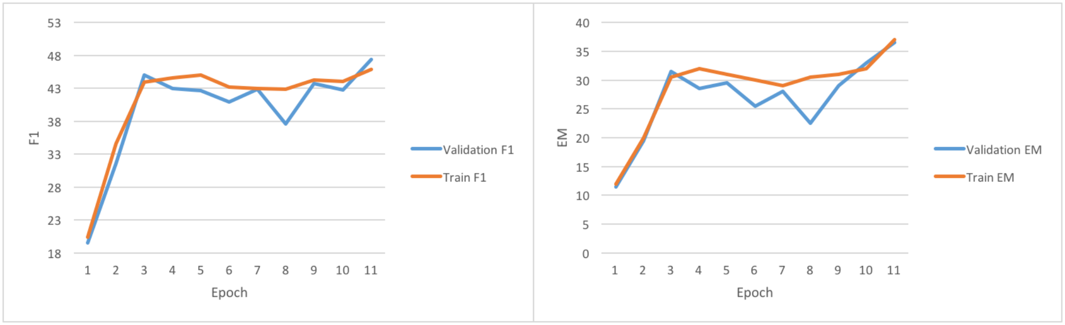

Figure 4: F1 / EM values during training over train and validation sets

5.3 Analysis

5.3.1 Prediction Narrowing

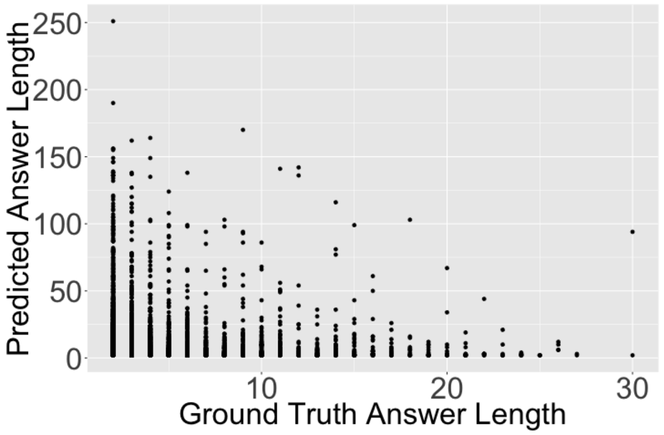

Figure 5: Comparison of ground truth answer lengths to predicted answer lengths on the dev set.

In Figure 5, we map our the length of our predicted answers in comparison to the length of their corresponding round truths. Two clear takeaways are readily apparent. The first is that our model generally produces answers that are significantly longer than the ground truth. The second is that, despite that, we are generally able to capture the ground truth in the span of our answer. This indicates that, perhaps, in future iterations, it would be fruitful to penalize the length of answers by incorporating a heuristic based on answer length into the loss function.

Ground Truth	Predicted
gold	golden anniversary with various gold-themed initiatives , as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals
American Football Conference	Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season . The American Football Conference

Figure 6: Examples of ground truth answers and predicted answers.

Further on, Figure 7 charts out the relationship between the lengths of questions, passages, ground truth answers, and the F1 scores our model achieves for them, respectively. We see that model performance is relatively invariant to the lengths of questions, passages, and ground truths, which validates the use of padding as an apt strategy rectifying a high variance in the lengths of questions, passages, and ground truths..

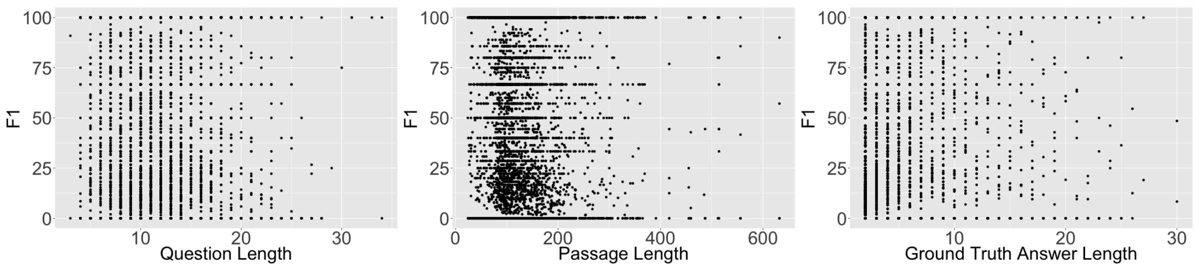

Figure 7: Comparison of passage, question, and ground truth answer lengths to corresponding F1 on the dev set.

5.3.2 Synonym Latching

Ground Truth	Predicted
diviners	haruspices
trophy hunting	poaching
everyday life	commonplace

Figure 8: Examples of single-word answers we predict using synonyms.

In many cases, the model struggles to differentiate synonyms, as evidenced by its tendency to predict one-word answers that are synonyms to one-word ground truths. This attachment to synonyms suggests that the model might be latching on to possible answers that are comprised of words that appear in both the question and in the passage, rather than pinpointing the synonymic answer. Likely due to the application of both relevancy scaling and the coattention mechanism in the encoder, the phenomenon suggests that the model attends too closely to questions, unable to generalize when predicting an answer.

6 Conclusion and Future Work

In this paper, we present a relevancy-scaled deep co-attentive network for question answering. Though its results leave much to be desired with a best-performing test-set F1 level of 43.15, the iterative process of experimentation that lead up to its generation yielded many interesting insights that can be leveraged in future implementations – be they lessons learned from failed optimization attempts (i.e. Gaussian Cross Entropy Loss), or more successful augmentations (i.e. Dynamic Programming Answer Mechanism). We hypothesize several angles for future work and improvement in the model.

To prevent the phenomenon of synonym latching, we propose either reducing relevancy scaling (using the mean of the relevancy scores instead of the max) or limiting dynamic co-attention. We believe we're weighting similarities between words in the question and passage too much and are not focusing enough on deciphering the logic between them. However, as we saw above, we cannot reduce it too much as we are still not doing a good job at differentiating what words are necessary to communicate the answer and what are superfluous neighboring words. To that end, we believe we need to create a more complex knowledge representation. Throughout the encoding process, we focus heavily on concatenation operations i.e. we concatenate forward and backward hidden states in the Bi-LSTM's and when creating our passage summaries in the co-attention operation. However, much like the act of merging knowledge isn't simply appending information, we believe including component-wise operations, such as the Hadamard product, in these steps will help us uncover more complex relationships between the passage and question.

Another future route involves a more novel reinvention of the neural question-answer scheme, a high-level description of which we attempt here. From the human cognition standpoint, there are two general cases that occur when attempting to answer questions based on passages. In the first, after understanding the question and passage, we immediately know what the answer is. However, for lengthy or logically complicated passages, this might not always be the case. In the second scenario, after reading the question and passage multiple times, we're not confident what the answer is. Yet, there are certain areas of the passage we know we can eliminate. After the elimination, we then redo a portion of our deductive reasoning on the remaining parts, iterating until we are confident enough in an answer (where the confidence threshold itself is also governed by some sort of probability distribution). We propose extending this logic to our model. For example, once we generate initial start index distributions, we extend another fully connected layer which acts a probability distribution on whether or not to stop iterating. If a certain threshold is passed, we compute the end-index distribution. However, if this threshold is not passed, we remove indices we are less confident about, re-calculate the co-attention, and repeat until our threshold is reached. The key challenge will be determining out how to make the above process differentiable.

Finally, we propose exploring incorporating dependency parsers in order to augment the representational power of our encoder. By incorporating the structural information gleaned by a dependency parser into our model, we believe that the model could be able to answer questions that span more complex semantic relationships in the passage.

References

[1] Answering Clinical Questions with Knowledge-Based and Statistical Techniques *Computational Linguistics* 2007

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text

[3] Claude Cordell Green. The application of theorem-proving techniques in question-answering systems *Doctoral Dissertation* 1969.

[4] Dekang Lin, Patrick Pantel. Discovery of Inference Rules for Question-Answering

[5] Cody Kwok, Oren Etzioni, Daniel S. Weld. Scaling Question Answering to the Web

[6] Claire Cardie, Vincent Ng, David Pierce, Chris Buckley. Examining the role of statistical and linguistic knowledge sources in a general-knowledge question-answering system *Proceedings of the Sixth Applied Natural Language Processing Conference* 2000

[7] Zhiguo Wang, Haitao Mi, Wael Hamza, Radu Florian. Multi-Perspective Context Matching for Machine Comprehension 13 Dec. 2016

[8] Dynamic Coattention Networks For Question Answering. Caiming Xiong, Victor Zhong, Richard Socher. Salesforce Research

[9] Jeffrey Pennington, Richard Socher, Christopher D. Manning. GloVe: Global Vectors for Word Representation 2014

[10] Sepp Hochreiter; Jrgen Schmidhuber (1997). "Long short-term memory"

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting

[12] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization 2014