Interspeech 2018
2-6 September 2018, Hyderabad

Gated Convolutional Neural Network for Sentence Matching

Peixin Chen, Wu Guo, Zhi Chen, Jian Sun, Lanhua You

National Engineering Laboratory for Speech and Language Information Processing, University of Science and Technology of China, Hefei, China

bubble@mail.ustc.edu.cn, guowu@ustc.edu.cn, zchen17@mail.ustc.edu.cn, sjian17@mail.ustc.edu.cn, lhyou@mail.ustc.edu.cn

Abstract

The recurrent neural networks (RNN) have shown promising results in sentence matching tasks, such as paraphrase identification (PI), natural language inference (NLI) and answer selection (AS). However, the recurrent architecture prevents parallel computation within a sequence and is highly time-consuming. To overcome this limitation, we propose a gated convolutional neural network (GCNN) for sentence matching tasks. In this model, the stacked convolutions encode hierarchical context-aware representations of a sentence, where the gating mechanism optionally controls and stores the convolutional contextual information. Furthermore, the attention mechanism is utilized to obtain interactive matching information between sentences. We evaluate our model on PI and NLI tasks, and the experiments demonstrate the advantages of the proposed approach in terms of both speed and accuracy performance.

1. Introduction

Many natural language tasks involve comparing two text sequences and determining the semantic relationship between them. For instance, in paraphrase identification, this comparison needs to determine whether two sentences are paraphrased [1, 2]. In natural language inference, the goal is to determine whether a hypothesis sentence can be inferred from a premise sentence [3, 4]. In answer selection, a question needs to be matched against several candidate answer sentences in order to identify the correct answer [5, 6].

With recent advances in neural network models, a straightforward approach for sequence modeling is to individually encode each sentence into a vector. Next, the two vectors are fed into a multi-layer perceptron together to predict the results [7]. The advantage of this framework is that the model is easier to train, and the sentence vectors can be used for visualization, sentence clustering and many other purposes [8]. However, there is no interaction between the two sentences during the encoding procedure, which is not sufficient to capture matching information of smaller units (such as words or phrases) between the two sentences. To overcome this disadvantage, many prevalent studies focus on the "compare-aggregate" framework, where the attention mechanism is employed on smaller units between the two sentences. In this type of framework, context representation layers are used to obtain the context-aware representations of words. Next, the context-aware vectors are compared and matched with attention mechanisms. These comparison results are aggregated to make the final decision [9, 10]. The "compare-aggregate" framework captures more interactive features between the two sentences, which leads to significant improvement. In state-of-the-art "compare-aggregate" frameworks, such as the Enhanced Sequential Inference Model (ESIM) [11] and the Bilateral Multi-perspective Matching (BiMPM) [12], the context representation layer and aggregation layer are all based on long short-term memory networks (LSTM) [13]. Despite the powerful sequence modeling ability, the recurrent structure depends on the computations of previous time steps, and the time complexity is linearly proportional to the sentence length.

As is well known, modern hardware is well suited to models that are highly parallelizable. An alternative approach is to capture context dependencies with convolutional neural networks (CNN), which enable parallelization within a sequence and are able to gain remarkable increased speed compared to recurrent networks. Furthermore, CNN provides a shorter path between two certain words. The context dependencies over a context window of n words require $O(n/k)$ convolutional operations with kernel width k, compared to $O(n)$ operations for recurrent networks. To take advantage of CNN, researchers have proposed the gated linear units (GLU) for language modeling, which implement a simplified gating mechanism over the convolution output [14]. More recently, the GLU has been introduced to sequence to sequence learning [15], and outperforms machine translation models based on gated recurrent units (GRU) and LSTM [16, 17]. Inspired by the success of GLU and gating mechanism of LSTM, we propose an improved gated convolutional architecture and apply it to the "compare-aggregate" framework for sentence matching tasks. The proposed gated convolutional network is equipped with output and forget gates. The output gate modulates the outputs of the current convolution layer. Meanwhile, the contextual information created by previous convolution layers are modulated by the forget gate and stored in the memory cells. The gating mechanisms further optimize the path through which information flows and lead to better performance for sentence matching tasks.

The rest of this paper is organized as follows. In section 2, we describe the model in detail. In section 3, we describe the datasets and experimental setup. Section 4 presents the experimental results and analysis. The summary and conclusions are given in section 5.

2. Model

In this section, we first describe the proposed gated convolutional architecture in detail and later apply it to the "compare-aggregate" framework. In addition, we describe the character and part-of-speech (POS) tagging features, which can be employed as auxiliary features in addition to word embeddings.

2.1. Gated Convolutional Neural Network

The GLU proposed in [14] are described as follows:

$$\mathbf{h}_i^L = (\mathbf{h}_i^{L-1}(c) * \mathbf{W} + \mathbf{b}) \odot \sigma(\mathbf{h}_i^{L-1}(c) * \mathbf{V} + \mathbf{c}) \quad (1)$$

10.21437/Interspeech.2018-70

where m,n are respectively the number of input and output feature maps, k is the kernel width, and L denotes the L^{th} convolution layer. $\mathbf{h}_i^{L-1}(c) \in \mathbb{R}^{m\times k}$ is the input, which comes from the hidden states of the previous layer or input word embeddings. $\mathbf{W} \in \mathbb{R}^{m\times k\times n}$ and $\mathbf{V} \in \mathbb{R}^{m\times k\times n}$ are convolution parameters, $\mathbf{b} \in \mathbb{R}^n$ and $\mathbf{c} \in \mathbb{R}^n$ are bias parameters. $*$ is the convolution operation, σ is the sigmoid function and $\odot$ is the element-wise multiplication. In GLU, the output of each layer is a linear projection $\mathbf{h}_i^{L-1}(c) * \mathbf{W} + \mathbf{b}$ modulated by the gate $\sigma(\mathbf{h}_i^{L-1}(c) * \mathbf{V} + \mathbf{c})$. Similar to LSTM, the gate multiply each element of the vector $\mathbf{h}_i^{L-1}(c) * \mathbf{W} + \mathbf{b}$ and control the information passed on in the hierarchy.

To better control the path through which information flows in the hierarchical structure, we adopt the concepts of output and forget gates in our convolutional neural network. The expressions of the new GCNN proposed in this paper are described as follows:

$$\begin{aligned}
\mathbf{o}_i^L &= \sigma(\mathbf{h}_i^{L-1}(c) * \mathbf{W}_o + \mathbf{b}_o) \\
\mathbf{f}_i^L &= \sigma(\mathbf{h}_i^{L-1}(c) * \mathbf{W}_f + \mathbf{b}_f) \\
\mathbf{g}_i^L &= \tanh(\mathbf{h}_i^{L-1}(c) * \mathbf{W}_g + \mathbf{b}_g) \\
\mathbf{c}_i^L &= \mathbf{f}_i^L \odot \mathbf{c}_i^{L-1} + (1 - \mathbf{f}_i^L) \odot \mathbf{h}_i^{L-1} \\
\mathbf{h}_i^L &= \mathbf{o}_i^L \odot \mathbf{g}_i^L + \mathbf{c}_i^L
\end{aligned} \tag{2}$$

$\mathbf{W} \in \mathbb{R}^{m\times k\times n}$ and $\mathbf{b} \in \mathbb{R}^n$ are learned parameters. $\mathbf{o}_i^L$, $\mathbf{f}_i^L$ and $\mathbf{c}_i^L$ are respectively the output gate, forget gate and memory cell for the i^{th} word in layer L. The dimensions of $\mathbf{h}_i^{L-1}$ and $\mathbf{c}_i^{L-1}$ must be equal. If this equality is not observed, we can perform a linear projection to match the dimensions as follows:

$$\mathbf{c}_i^L = \mathbf{f}_i^L \odot \mathbf{c}_i^{L-1} + (1 - \mathbf{f}_i^L) \odot (\mathbf{W}_p \mathbf{h}_i^{L-1}) \tag{3}$$

The memory cell of LSTM is shared by all elements in the sequence and is updated along with time. Different from that of LSTM, each element has its own memory cell in GCNN, and the memory cell is updated along with layers. The memory optionally stores contextual information created by all previous convolution layers. The forget gate $\mathbf{f}_i^L$ controls what information from the old memory cell is going to be thrown away and what new information is going to be stored in the current memory cell. $\mathbf{c}_i^L$ contains semantics of smaller granularity, while $\mathbf{g}_i^L$ contains semantics of larger granularity. For example, when the kernel width k is set to 3, $\mathbf{g}_i^2$ contains 5-gram features, while $\mathbf{c}_i^2$ contains trigram features from $\mathbf{h}_i^1$, as well as unigram features from $\mathbf{c}_i^1$. We expect the output $\mathbf{h}_i^2$ of the current layer to benefit from unigram, trigram and 5-gram information simultaneously, which are regulated by the forget and output gates. The memory cell can also be viewed as a modified residual learning, which simplifies the model training at the same time.

2.2. Convolutional Compare-Aggregate Framework

In this section, we apply the abovementioned GCNN to the "compare-aggregate" framework. Let $A = [a_1, ..., a_{l_a}]$ and $B = [b_1, ..., b_{l_b}]$ be a pair of input sentences of length l_a and l_b , respectively, and $y \in Y$ be the label representing the relationship between A and B. For paraphrase identification task, Y= $\{0, 1\}$, where $y = 1$ means A and B are paraphrase of each other, and $y = 0$ otherwise. For natural language inference task, $Y = \{entailment, contradiction, neural\}$, where entailment means the hypothesis sentence B can be inferred from the premise sentence A, contradiction means the premise B cannot be true condition on A, and $neutral$ means A and B are irrelevant to each other. The convolutional "compare-aggregate" framework consists of five layers: word embedding layer, context representation layer, comparison layer, aggregation layer, and prediction layer.

2.2.1. Word embedding layer

In this layer, each word in A and B is represented with an embedding of the fixed-dimensional vector, which can be initialized with word2vec [18] or GloVe [19]. The output of this layer are two sequences of word vectors $\mathbf{a} = [\mathbf{a}_1, ..., \mathbf{a}_{l_a}]$ and $\mathbf{b} = [\mathbf{b}_1, ..., \mathbf{b}_{l_b}]$.

2.2.2. Context representation layer

The purpose of this layer is to capture the contextual information. Let $\widetilde{\mathbf{a}}_i$ denote the output state generated by the multi-layer GCNN for the i^{th} element over the input sequence $\mathbf{a}$ as follows:

$$\begin{aligned}
\widetilde{\mathbf{a}}_i &= \mathrm{GCNN}(\mathbf{a}, i), \forall i \in [1, ..., l_a] \\
\widetilde{\mathbf{b}}_j &= \mathrm{GCNN}(\mathbf{b}, j), \forall j \in [1, ..., l_b]
\end{aligned} \tag{4}$$

2.2.3. Comparison layer

In this layer, the context representation of each word in one sentence is compared against all words in the other sentence. In other words, the attention mechanisms are utilized to compute the alignment score between two elements from $\widetilde{\mathbf{a}}$ and $\widetilde{\mathbf{b}}$. MLP attention [20] and dot-product attention [21] are the two most commonly used attention mechanisms, where the latter is faster and more memory-efficient than the former. We adopt dot-product attention to compute the attention weight between each pair of $\widetilde{\mathbf{a}}_i$ and $\widetilde{\mathbf{b}}_i$ as follows:

$$e_{ij} = \widetilde{\mathbf{a}}_i^T \widetilde{\mathbf{b}}_j \tag{5}$$

These attention weights are normalized to compute the attention vectors as follows:

$$\begin{aligned}
\boldsymbol{\alpha}_i &= \sum_{j=1}^{l_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_b} \exp(e_{ik})} \widetilde{\mathbf{b}}_j, \forall i \in [1, ..., l_a] \\
\boldsymbol{\beta}_j &= \sum_{i=1}^{l_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_a} \exp(e_{kj})} \widetilde{\mathbf{a}}_i, \forall j \in [1, ..., l_b]
\end{aligned} \tag{6}$$

where the attention vector $\boldsymbol{\alpha}_i$ selects the content semantics from $\widetilde{\mathbf{b}}_j$ that is relevant to $\boldsymbol{\alpha}_i$, and vice versa for $\boldsymbol{\beta}_i$. The absolute difference and element-wise product are then performed to obtain the comparison vectors [11, 22]:

$$\begin{aligned}
\mathbf{m}_{a,i} &= [\widetilde{\mathbf{a}}_i; \boldsymbol{\alpha}_i; |\widetilde{\mathbf{a}}_i - \boldsymbol{\alpha}_i|; \widetilde{\mathbf{a}}_i \odot \boldsymbol{\alpha}_i] \\
\mathbf{m}_{b,j} &= [\widetilde{\mathbf{b}}_j; \boldsymbol{\beta}_j; |\widetilde{\mathbf{b}}_j - \boldsymbol{\beta}_j|; \widetilde{\mathbf{b}}_j \odot \boldsymbol{\beta}_j]
\end{aligned} \tag{7}$$

The attention operation, together with the absolute difference and element-wise product operations, brings in interactive matching information between the two sentences.

2.2.4. Aggregation layer

In this layer, we aggregate the two sequences of comparison vectors into a fix-dimensional vector $\mathbf{v}$. We apply another multi-layer GCNN to the two sequences $\{\mathbf{m}_{a,i}\}_{i=1}^{l_a}$ and $\{\mathbf{m}_{b,j}\}_{j=1}^{l_b}$ individually and obtain the outputs $\mathbf{v}_a = [\mathbf{v}_{a,1}, ..., \mathbf{v}_{a,l_a}]$ and $\mathbf{v}_b = [\mathbf{v}_{b,1}, ..., \mathbf{v}_{b,l_b}]$. Then, we compute both the average and max pooling vectors as well as concatenating them to form the vector $\mathbf{v}$ [11]:

$$\mathbf{v} = [\max(\mathbf{v}_a); \mathrm{ave}(\mathbf{v}_a); \max(\mathbf{v}_b); \mathrm{ave}(\mathbf{v}_b)] \tag{8}$$

where max() is the element-wise maximum, and ave() computes the mean of vectors.

2.2.5. Prediction layer

The purpose of this layer is to predict the probability distribution $P(y|A, B)$. To this end, we feed $\mathbf{v}$ into a multi-layer perceptron (MLP) classifier. The MLP has a hidden layer with ReLU activation and an output layer with the softmax function. The entire model is trained end-to-end with cross-entropy loss.

2.3. Character and Part-of-Speech Tagging Features

To further improve the learning and expressing ability of the model, we can also incorporate the character-composition features and part-of-speech tagging features into the input. The character-composition vector is expected to encode more morphological information for a word [23], and the POS tagging feature is expected to provide syntactic information. Meanwhile, the word embeddings initialized by word2vec or GloVe encode more semantic information. The semantic, morphological and syntactic information are complementary to each other, and combinations of these different information is expected to improve the model performance.

Unlike previous work where the character-composition vectors are obtained by conventional CNN or LSTM networks [12, 24], we employ the GCNN. $[c_1, ..., c_n]$ denote the characters within a word, we first obtain the vectors $[\mathbf{c}_1, ..., \mathbf{c}_n]$ by embedding the characters into fixed-dimensional character embeddings, which are initialized randomly and updated during the training. Next, a GCNN with H layers is employed on them to obtain the outputs $[\mathbf{c}_1^H, ..., \mathbf{c}_n^H]$. Finally, the max pooling operation is performed to obtain the character-composition vector:

$$\mathbf{c} = \max(\mathbf{c}_1^H, ..., \mathbf{c}_n^H) \tag{9}$$

POS tagging is produced by lexical parser tools and subsequently mapped to one-hot vector for each word in a sentence. The character-composition vector $\mathbf{c}$ and the one-hot POS vector are concatenated with the word embedding as the input features.

3. Experimental Setup

3.1. Datasets

We evaluate our model on NLI and PI tasks, which are conducted on the MultiNLI dataset[1] and the Quora question pair dataset[2] respectively. MultiNLI [25] has 433k sentence pairs annotated with textual entailment labels. It covers a range of genres of spoken and written text, and supports a distinctive cross-genre generalization evaluation. Half of the dev/test genres appear in the training set while the remainder do not, creating in-domain (matched) and cross-domain (mismatched) dev/test sets. Since test set labels are not provided, the test is performed on Kaggle.com[3]. The Quora question pair dataset contains over 400k question pairs, and each question pair is annotated with a binary value indicating whether the two questions are paraphrases of each other. We use the same train/dev/test split as mentioned in previous research [12].

[1] https://www.nyu.edu/projects/bowman/multinli/

[2] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[3] In-domain (matched) leaderboard:
https://www.kaggle.com/c/multinli-matched-open-evaluation/leaderboard;
Cross-domain(mismatched) leaderboard:
https://www.kaggle.com/c/multinli-mismatched-open-evaluation/leaderboard

Table 1: *MultiNLI results*

Models	Matched Test Acc	Mismatched Test Acc	Time(s) /Epoch
CBOW [25]	64.8	64.5	–
BiLSTM [25]	66.9	66.9	–
ESIM [25]	72.3	72.1	–
CNN	75.6	74.5	1215
GLU	76.3	75.2	1650
ESIM (our implement)	76.8	75.5	9610
GCNN	76.6	75.8	2412
GCNN (char)	77.0	76.0	4020
GCNN (char+POS)	77.4	76.4	4080

Table 2: *Quora question pair dataset results*

Models	Test Acc	Time(s)/Epoch
BiMPM [12]	88.17	–
DIIN [24]	89.06	–
CNN	87.92	660
GLU	88.32	692
ESIM (our implement)	87.65	4038
GCNN	88.66	870
GCNN (char)	88.94	1740
GCNN (char+POS)	89.35	1770
GCNN - forget gate	88.33	690
GCNN - output gate	88.35	702

3.2. Training

We implement our models on Tensorflow [26] and train them on an Nvidia GeForce GTX 1080 GPU. We initialize the word embeddings with 300-dimensional GloVe word vectors pretrained from 840B Common Crawl corpus [19]. Out-of-vocabulary (OOV) words are randomly initialized. We initialize each character as a 40-dimensional vector and compose the 100-dimensional character-composed vector for each word with a 2-layer GCNN. The POS tagging feature for each word is a 47-dimensional one-hot vector. Our model has 4 convolution layers in the context representation layer and 2 layers in the aggregation layer. The hidden size of each convolution layer is set to 300. The sequence length is set as a hard cutoff on all experiments, with 60 for premise sentences and 30 for hypothesis sentences on MultiNLI and 40 for sentences on the Quora dataset. The Adam algorithm [27] is adopted as optimizer. The initial learning rate is 0.0004, and the batch size is 64. For all the experiments, we pick the model which works best on the dev set, and then evaluate it on the test set.

4. Results and Model Analysis

The experiment results are listed in Table 1 and Table 2, where GCNN is the proposed model with only word embedding inputs, GCNN (char) is additionally equipped with character features on the basis of GCNN, and GCNN (char+POS) is additionally equipped with both character and part-of-speech tagging features. We also implement the ESIM on Tensorflow. In Table 1, we also list the results reported in the dataset paper [25]. From Table 1, it can be observed that the training speed of GCNN is clearly faster than that of ESIM. Furthermore, GCNN performs nearly equal to ESIM in terms of accuracy on MultiNLI dataset. The accuracy is further improved when in-

corporating additional character features into the inputs, which shows the effectiveness of character features. Similar conclusions can be drawn for part-of-speech tagging features. To better demonstrate the advantage of GCNN, we also replace the GCNN with conventional CNN; namely, we replace equation (2) with $\mathbf{h}_i^L = \mathrm{ReLU}(\mathbf{h}_i^{L-1}(c) * \mathbf{W}_c + \mathbf{b}_c) + \mathbf{h}_i^{L-1}$. It can be found that CNN is inferior to GCNN in terms of accuracy. We also replace the GCNN with GLU. It can be found that GCNN performs better than GLU in terms of accuracy.

In Table 2, the results again demonstrate the superiority of our approach. Furthermore, GCNN performs better than ESIM in terms of accuracy on Quora dataset. We also compare our model with BiMPM [12] and Densely Interactive Inference Network (DIIN) [24], which are published state-of-the-art results on this dataset. BiMPM also employs the characters in addition to word embeddings, where the character-composed vectors are constructed via LSTM. Table 2 shows that GCNN outperforms BiMPM even without character features. DIIN solve sentence matching tasks by extracting semantic features from the interaction tensor with DenseNet [28]. DIIN also incorporates character features and one-hot POS tagging features into the inputs. It can be observed that our GCNN (char+POS) outperforms DIIN on the Quora dataset. To further analyze the importance of forget and output gates, we also show the ablation performance of GCNN, with only word embedding inputs. In "GCNN - forget gate", we remove the forget gate, namely, replacing $\mathbf{h}_i^L = \mathbf{o}_i^L \odot \mathbf{g}_i^L + \mathbf{c}_i^L$ with $\mathbf{h}_i^L = \mathbf{o}_i^L \odot \mathbf{g}_i^L + \mathbf{h}_i^{L-1}$. The performance drops to 88.33% with this replacement. In "GCNN - output gate", we replace it with $\mathbf{h}_i^L = \mathbf{g}_i^L + \mathbf{c}_i^L$. We adopt ReLU activation for $\mathbf{g}_i^L$ instead of tanh in this replacement, since ReLU performs better than tanh when there is no output gate. It shows that the performance drops to 88.35% without the output gate. The ablation results show that both the forget and output gates contribute to the model performance.

5. Conclusion

In this paper, we propose a new gated convolutional network and apply it to the "compare-aggregate" framework. Compared to recurrent neural networks, the convolution operations allow parallel computation within a sequence. Meanwhile, the hierarchical architecture makes it easier to capture long-range dependencies. Furthermore, the gating mechanism makes the network able to optionally allow information flow through the hierarchical structure. Experiments on standard benchmark datasets show the effectiveness of our model in terms of both accuracy and computation speed.

6. Acknowledgements

This work was partially funded by the National Key Research and Development Program of China (Grant No. 2016YFB100 1303. Besides, we would like to thank the anonymous reviewers for their valuable comments and suggestions.

7. References

[1] N. Madnani, J. Tetreault, and M. Chodorow, "Re-examining machine translation metrics for paraphrase identification," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2012, pp. 182–190.

[2] W. Yin and H. Schütze, "Convolutional neural network for paraphrase identification," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 901–911.

[3] M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli, "Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment," in *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, 2014, pp. 1–8.

[4] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015, pp. 632–642.

[5] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman, "Deep learning for answer sentence selection," *NIPS deep learning workshop*, 2014.

[6] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, "Movieqa: Understanding stories in movies through question-answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4631–4640.

[7] M. Feng, B. Xiang, M. R. Glass, L. Wang, and B. Zhou, "Applying deep learning to answer selection: A study and an open task," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 813–820.

[8] Z. Wang, H. Mi, and A. Ittycheriah, "Semi-supervised clustering for short text via deep representation learning," *CoNLL*, 2016.

[9] S. Wang and J. Jiang, "Learning natural language inference with lstm," *In Proceedings of the Conference on the North American Chapter of the Association for Computational Linguistics*, 2016.

[10] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[11] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced lstm for natural language inference," in *Proc. ACL*, 2017.

[12] Z. Wang, W. Hamza, and R. Florian, "Bilateral multi-perspective matching for natural language sentences," in *26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017, pp. 4144–4150.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," *arXiv preprint arXiv:1612.08083*, 2016.

[15] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *arXiv preprint arXiv:1705.03122*, 2017.

[16] R. Sennrich, B. Haddow, and A. Birch, "Edinburgh neural machine translation systems for wmt 16," *arXiv preprint arXiv:1606.02891*, 2016.

[17] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

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

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[21] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *In EMNLP*, 2015.

[22] L. Mou, R. Men, G. Li, Y. Xu, L. Zhang, R. Yan, and Z. Jin, "Natural language inference by tree-based convolution and heuristic matching," in *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016, pp. 2315–2325.

[23] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models." in *AAAI*, 2016, pp. 2741–2749.

[24] Y. Gong, H. Luo, and J. Zhang, "Natural language inference over interaction space," *arXiv preprint arXiv:1709.04348*, 2017.

[25] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.

[26] 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.

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

[28] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," *arXiv preprint arXiv:1608.06993*, 2016.