2018 IEEE International Conference on Big Data and Smart Computing

Contextual-CNN: A Novel Architecture Capturing Unified Meaning for Sentence Classification

Joongbo Shin, Yanghoon Kim, Seunghyun Yoon and Kyomin Jung
Dept. of *Electrical and Computer Engineering, Seoul National University*, Seoul, Korea
{jbshin, ad26kr, mysmilesh, kjung}@snu.ac.kr

***Abstract*—In this paper, we focus on the architecture of the convolutional neural network (CNN) for sentence classification. For understanding natural language, context in the sentence is important information for grasping the word sense. However, traditional CNN's feed-forward architecture is insufficient to reflect this factor. To solve this limitation, we propose a contextual CNN (C-CNN) for better text understanding by adding recurrent connection to the convolutional layer. This architecture helps C-CNN units to be modulated over time with their neighboring units, thus the model integrates word meanings with surrounding information within the same layer. We evaluate our model on sentence-level sentiment prediction tasks and question categorization task. The C-CNN achieves state-of-the-art performances on fine-grained sentiment prediction and question categorization.**

***Index Terms*—deep learning, convolutional neural network, natural language processing, sentence classification**

I. Introduction

Convolutional neural networks (CNNs) have achieved great successes in computer vision, and this prosperity has been extended to natural language processing (NLP) in recent years. Without using well-studied classical hand-crafted features in natural language domain, CNN based models have shown the state-of-the-art performances on sentence classification tasks [1]–[4]. These achievements are mainly due to the CNN's power of extracting local features from the data by using convolution layers and accumulating global information by building hierarchical structures.

Contextual information in a sentence is significant to disambiguate the meaning of words. For instance, the following two sentences "I slept deeply at night." and "I slept deeply halfway through this movie." have the same phrase "slept deeply" which should be recognized differently in terms of sentimental meaning. While CNN's hierarchical layers can deal with low-to-high level information, it has limitations in capturing contextual meaning of words within the whole sentence because its architecture relies on the feed-forward hierarchical path, particularly in the inference stage.

K. Jung is with the Department of Electrical and Computer Engineering, ASRI, Seoul National University, Seoul, Korea. This work was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2016M3C4A7952587), the Ministry of Trade, Industry & Energy(MOTIE, Korea) under Industrial Technology Innovation Program(No.10073144), and the Brain Korea 21 Plus Project.

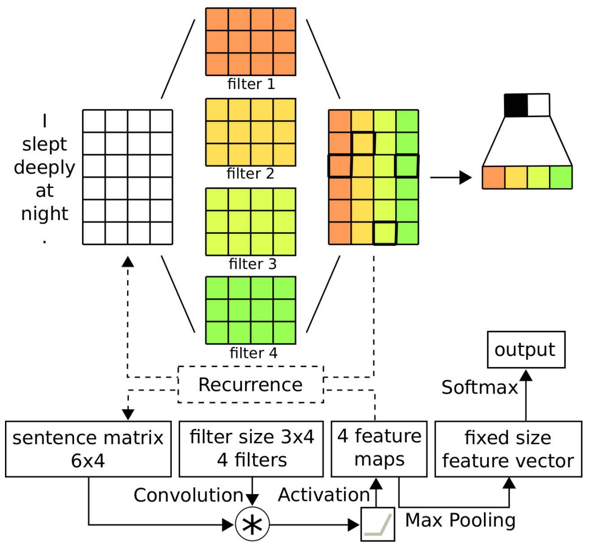

Fig. 1. Illustration of an C-CNN architecture for assigning the sentence to the one of two labels. The color is assigned to the units corresponding to the filter, and bold boxes in features are pooled units. If one omitting the dashed arrows for recurrent connection, this figure shows one layer CNN model for sentence classification.

For better understanding sentences, we propose a novel CNN-based scheme that properly integrates feature extraction and context modulation. An example of the proposed model, contextual CNN (C-CNN for convenience) is illustrated in Fig. 1. The key module of our model consists of original convolution and additional recurrence function. This architecture allows C-CNN units to be updated according to their neighboring units, hence the model secure the ability of context modulation within the same layer. At the first layer of C-CNN, right after the sentence matrix, feature maps will represent a set of new word vectors updated by original words, taking more peripheral contexts into account over time. Moreover, non-linear interactions of words within a local context can be obtained by recurring of the convolution with non-linear activation function. It is worth noting that the model's revising process at word level is also available in the inference stage, which is important to understand the text as human do.

We evaluate the proposed model on sentence-level text classification tasks, including standard sentiment prediction benchmark Stanford Sentiment Treebank and question categorization dataset TREC. The C-CNN achieves state-of-the-art accuracy 52.3% on the fine-grained sentiment prediction task (5-class) and 95.2% on the TREC question categorization task (6-class). To verify the C-CNN's performance over our CNN baselines, we conduct additional experiments on the sentiment prediction dataset. Experimental results demonstrate that the proposed model has an advanced architecture for NLP tasks.

II. Related Works

A. CNN-based models for text understanding

Many prior CNN-based works have been proposed for the sentence-level text classification tasks, and they have achieved excellent performances without hand-crafted features. Propagating extracted features from convolution layer to the logistic regression layer needs an encoding process which converts a sentence of arbitrary length to a fixed sized vector. A common approach for this process is max-pooling over all features (1-max pooling for convenience) detected by a sliding convolution filters on a given sentence [1], [2]. However, stacking convolution layers, which is essential to represent a hierarchical structure of a sentence, cannot be obtained with 1-max pooling strategy.

To overcome this limitation, k-max pooling is proposed, which selects k most active features with preserved order. To make parameter k sensitive to the length of the sentence and the depth of the network, a dynamic k-max pooling is proposed in [3]. These pooling strategies allow CNN architecture to have multiple convolution layers for extracting high-level abstract features. In order to get higher classification accuracies, extensive experiments with different model architectures are conducted in [4]. They adopt hybrid word embedding layer by utilizing pretrained word embeddings like GloVe [5] to resolve out-of-vocabulary problem. They also apply different-sized filters for better phrase detection, and some tricks for pretraining the networks.

In [6], CNN architecture is revised to get non-linear interactions between words and non-sequential convolution. They try to capture the non-consecutive n-grams, and achieve good performance in several text classification tasks.

Those CNN-based models have tried to better represent a sentence, however, none of them applies context modulation within a sentence while reading the text.

B. Recurrent and Convolutional Neural Networks

Our study of recurrent connections in convolutional layer is inspired by a recently proposed model which shows better performance in image understanding [7]. However, we applied recurrent connections between convolution layers for capturing expanded contextual meaning of the text, thus building a novel architecture for sentence classification task.

It is worthwhile to mention that there is another work for capturing contextual meaning of words by using recurrent and convolutional neural networks for text classification [8]. They used two kinds of neural networks separately, however, we combined both recurrence and convolution into one layer that is recurrently applying convolution operation.

III. Model

A. CNN for Sentence Classification

In this section, we explain the elements of CNN for sentence classification as preliminaries for our study. Fig. 1 without dashed arrows shows an example of one layer CNN architecture.

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the input sentence matrix, which is concatenation of n word vectors of dimension d (*e.g.*, $n = 6$ and $d = 4$ in Fig. 1). In representing sentence using CNN, there is a common problem that input sequence length is various while the network generally needs fixed size output. Here we use the most standard remedy of fixing the input sequence to the length n_0: Any word exceeding length n_0 is ignored, and sentences shorter than the length are filled with all-zero vectors as well as out-of-vocabulary words.

A convolutional layer (CL) computes 1-D convolution over the sentence matrix with its k filters (or kernels) denoted by the matrix $\mathbf{W}^c_j \in \mathbb{R}^{m \times d}$, which is a j^{th} filter with region size m (*e.g.*, $k = 4$ and $m = 3$ in Fig. 1). An output of the layer located at i^{th} index on the j^{th} feature map is given by

$$z_{i,j} = f(\mathbf{W}^c_j \cdot \mathbf{X}_{[i]} + b^c_j), \qquad (1)$$

where $\mathbf{X}_{[i]} \in \mathbb{R}^{m \times d}$ is a sub-matrix of the sentence matrix centered at i^{th} position, b_j is the bias term, and f is the non-linear activation function such as rectified linear unit (ReLU). In this temporal convolution, we use stride 1 as other CNN based sentence classifiers do and pad zero vectors to the input, so the output of the convolution is $\mathbf{Z} \in \mathbb{R}^{n \times k}$.

To build up the layers for the hierarchical modeling, 1-D max pooling operation is applied to each feature map corresponding to particular filter with the $s \times 1$ pooling size, hence the resulting output will be $\mathbf{Z}' \in \mathbb{R}^{\frac{n}{s} \times k}$ (this operation is not represented in Fig. 1). Stacking L CLs with 1-D max pooling allows the CNN to capture more abstract meaning of the sentence. For the last convoluional layer right before the logistic regression, we apply 1-max pooling over each feature map, then we get a vector representing sentence $[y_1, \ldots, y_k] = \mathbf{y} \in \mathbb{R}^k$, where $y_j = \max(\mathbf{z}_{:,j})$ [1] (in Fig. 1, bold boxes represent maximum value in each feature map). After the 1-max pooling, we get the fixed size vector representation of the input sentence.

The sentence representation is forwarded to logistic regression layer for assigning it to one of C categories (*e.g.*, $C = 2$ in Fig. 1) with weight $\mathbf{W}^h \in \mathbb{R}^{k \times C}$ and bias $\mathbf{b}^h$. Following standard practices, our model is trained by minimizing the cross-entropy loss of predictions on a given training data.

B. C-CNN for Sentence Classification

The key module of C-CNN is the contextual convolutional layer (C-CL) having recurrent structure. The module repeatedly computes convolution over the sentence matrix thereby its output updates itself over time. An output unit located at i^{th} index of the j^{th} feature map at time step t of the layer is given by:

$$z_{i,j}(t) = f(x_{i,j} + \mathbf{W}^c_j \cdot \mathbf{Z}_{[i]}(t-1) + b^c_j), \qquad (2)$$

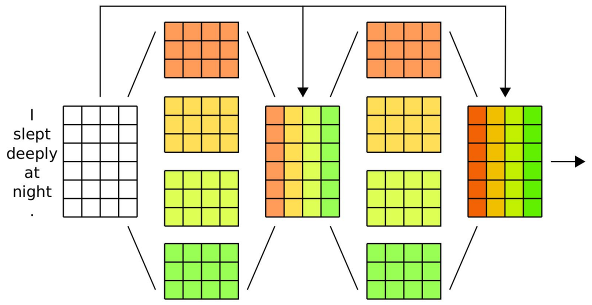

Fig. 2. The module of C-CNN is unfolded for $T = 2$ time steps.

where $\mathbf{Z}_{[i]}(t-1) \in \mathbb{R}^{m \times d}$ is a sub-matrix of the recurrent input, which is the previous output of the same layer, and the other variables and parameters are just the same as those in the previous section. The output of C-CL will be forwarded to the next layer after the finite iteration step T, and the other modules like pooling and logistic regression form the overall architecture together.

This lateral connection facilitates context modulation within the same layer, which is key idea of this work. Equation (2) and Fig. 2 represent the dynamic behavior of the proposed module. One of interesting points is that a state of the C-CL evolves over time while the input sentence matrix is static. The C-CL reinterprets meaning of the words from their surroundings, which cannot be obtained from the original CNN. The context region for updating current position depends on filter size m and predefined iteration step T: $(m-1)T+1$. After passing through the non-linear activation function and summing with the input, the C-CL attains non-linear relation of the words thus enriching meaning of the words.

We now describe the additional components of the traditional CL in our C-CNN maintaining the other elements in the previous section. Fig. 1 with recurrence shows an example of one layer C-CNN architecture.

By excluding the first term in (2), context modulation also can be formed

$$z_{i,j}(t) = f(\mathbf{W}^c_j \cdot \mathbf{Z}_{[i]}(t-1) + b^c_j).$$

We named this module as recurrent convolutional layer (R-CL), and the network composed of R-CLs as recurrent CNN (R-CNN). This is equivalent to (1) for $t = 1$ when $\mathbf{Z}$ is initialized as input matrix. Also, R-CNN with T iteration steps can be shown as T layers CNN without 1-D max pooling layer having shared filter weights across the layers.

For all kinds of convolutional layer, after computing the convolution, the local response normalization tailored for our task is used in all implementations in order to prevent the states from exploding:

$$g(z_{i,j}(t)) = \frac{z_{i,j}(t)}{\left(K + \alpha \sum_{j'=\max(0,j-(N-1)/2)}^{\min(k,j+(N-1)/2)} z_{i,j'}(t)^2\right)^{\beta}},$$

where the sum runs over N adjacent feature maps, and the constants K, α, and β are hyper-parameters of controlling the amplitude of normalization.

IV. Experiment

A. Datasets

We evaluate the proposed model on Sentiment Stanford Treebank benchmark (SST)[1]. SST-5 consists of movie review with find-grained labels (very positive, positive, neutral, negative, very negative). Following the previous works, we use standard 8544/1101/2210 split for training/development/testing, and also use the phrase-level labeled dataset for training. SST-2 is the binary version (positive, negative) of this benchmark obtained by ignoring neutral label and merging each polarity, and the resulting split is 6920/872/1821. Development data is used for hyper-parameter tuning, and also used in early stopping for preventing from over-fitting.

We also test our model on TREC question classification dataset (TREC)[2], which has a question set belonging to six major categories (abbreviation, entity, description, human, location, numeric). TREC consists of 5452/500 dataset for training/testing. To generate development dataset, we randomly sampled 452 from the training set.

For all datasets, we use lower-cased words with splitting by space and use accuracy as the metric.

B. Implementation Details

1) Overall architecture: $\{\mathbf{E}, \mathbf{W}^c_{1:L}, \mathbf{b}^c_{1:L}, \mathbf{W}^h, \mathbf{b}^h\}$ are the weight parameters that can be trained, where $\mathbf{E} \in \mathbb{R}^{V \times d}$ is lookup table for word vector representation with vocabulary size V and embedding dimension d, and $\mathbf{W}^c_{1:L}$ indicates all weights for L convolutional layers. For word representation, we use the publicly available 300-dimensional GloVe trained on the Common Crawl with 42B tokens [5], hence our embedding dimension $d = 300$. Word embedding is normalized to unit norm and is fixed in the experiments without fine-tuning. For sentence matrix, s_0 is chosen as maximum length of the sentence in each dataset.

To verify the ability of proposed model, we limit the search on hyper-parameter space as the iteration number $T \in \{1, 2\}$ and the number of the layers $L \in \{1, 2, 3\}$. Also, the same iteration is given to each layers, and every layer having the same k filters. For fair comparison, we set all our implementations having same number of weight parameters. With the condition $d = k$ and $T = 1$, R-CNN is identical to CNN, hence we set the number of feature maps $k = 300$. The rest hyper-parameters of model are set as $m = 3$, $s = 3$, $K = 1$, $\alpha = 0.001$, $\beta = 0.75$, and $N = 41$.

2) Training the model: In our implementation, the models are trained using Adam [9] updates rule in combination with the BPTT algorithm with gradient clipping over shuffled mini-batch for minimizing the cross-entropy loss. The initial

[1]http://nlp.stanford.edu/sentiment/
[2]http://cogcomp.cs.illinois.edu/Data/QA/QC

TABLE I
COMPARISON BETWEEN C-CNN AND OUR BASELINES ON SST. 'ITER' INDICATES THE ITERATION STEPS AND R-CNN-ITER1 REPRESENTS OUR IMPLEMENTATION OF CNN.

Model	SST-5			SST-2		
R-CNN-iter1 layer1/2/3	47.8	50.9	50.6	86.9	87.9	88.2
R-CNN-iter2 layer1/2/3	49.7	51.0	51.1	88.5	88.6	88.0
C-CNN-iter1 layer1/2/3	48.2	51.5	**52.3**	87.1	88.2	**89.2**
C-CNN-iter2 layer1/2/3	49.9	52.1	52.1	88.4	88.9	88.8

learning rate is set to 0.001 with the decay factor of 0.1 which will be applied every 5 epochs. We use Xavier initializer [10] for model initialization. During training, dropout is used after each convolutional layer with probability 0.25. We also use l_2 regularization with weight $1e-5$ for all datasets.

C. Results

1) Comparison with Baseline Models: We analyze the proposed models by comparing C-CNN with our baselines R-CNN and CNN on SST benchmark. For fair comparison, we set our implementations to have the same number of weight parameters. Varying the iteration steps and the number of layers, the test accuracies of each implementation of the models are shown in Tab. I. R-CNN with 1 iteration is identical to CNN in our setting, and we do not report the results for $T \geq 3$ because there is no advantage in accuracy.

When comparing line 1 with line 2 in Tab. I, we can verify the advantage of the recurrent connection in the convolutional layer as most results of R-CNN are better than those of CNN. Mostly, C-CNN is better than R-CNN ({line 3 and 4} vs line 2 in Tab. I), demonstrating that the combination of the static input and the dynamic output helps the model overcome the over-fitting problem. Line 2 and 4 show that sentence-level understanding may be hampered by stacking more than 2 layers at 2 iteration steps, and possible solution is to use different iteration in each C-CL. These results support that C-CNN has a better architecture over CNN for sentence classification.

2) Comparison with State-of-the-art Models: We compare our C-CNN with other models on sentence-level text classification tasks. Each block shows non-CNN models, CNN-based models, and our models from top to bottom, respectively.

Our C-CNN achieves better performances than state-of-the-art models on SST-5 and TREC, and shows competitive accuracy on SST-2. Tab. II shows that C-CNN outperforms all other CNN based models except for MVCNN on SST-2, which is slightly better than ours. MVCNN utilize extensive sources such as four different-sized filters, five pretrained word embeddings as well as pretraining the network to get the highest score.

To our knowledge, there have been limited trials on classifying TREC dataset with neural networks. This is due to the small size of the dataset, and it makes us hard to escape over-fitting problem. C-CNN shows slightly better result than the state-of-the-art method SVM [11], and significantly outperforms other CNN based models.

TABLE II
COMPARISON BETWEEN C-CNN WITH THE STATE-OF-THE-ART MODELS. THE FIRST BLOCK LISTS NON-CNN MODELS, AND THE SECOND BLOCK IS FOR CNN-BASED MODELS. THE LAST BLOCK IS OUR MODEL.

Model	SST-5	SST-2	TREC
SVM [11]	-	-	95.0
PVec [12]	48.7	87.8	-
RNTN [13]	45.7	85.4	-
T-LSTM [14]	51.0	88.0	-
DMN [15]	52.1	88.6	-
CNN [2]	48.0	87.2	93.6
DCNN [3]	48.5	86.8	93.0
MVCNN [4]	49.6	**89.4**	-
T-CNN [6]	51.2	88.6	-
C-CNN-iter1 layer1	48.2	87.1	90.8
C-CNN-iter1 layer2	51.5	88.2	94.4
C-CNN-iter1 layer3	**52.3**	89.2	**95.2**

V. CONCLUSION

We proposed a novel CNN architecture for sentence classification. By combining recurrent connections with convolutional layer, our network effectively integrated feature extraction and context modulation within the same layer. Experimental results demonstrated that the proposed model has an better CNN architecture for NLP tasks.

REFERENCES

[1] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.
[2] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.
[3] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.
[4] W. Yin and H. Schütze, "Multichannel variable-size convolution for sentence classification," *arXiv preprint arXiv:1603.04513*, 2016.
[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation."
[6] T. Lei, R. Barzilay, and T. Jaakkola, "Molding cnns for text: non-linear, non-consecutive convolutions," *arXiv preprint arXiv:1508.04112*, 2015.
[7] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3367–3375.
[8] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification." in *AAAI*, 2015, pp. 2267–2273.
[9] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[10] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." in *Aistats*, vol. 9, 2010, pp. 249–256.
[11] J. Silva, L. Coheur, A. C. Mendes, and A. Wichert, "From symbolic to sub-symbolic information in question classification," *Artificial Intelligence Review*, vol. 35, no. 2, pp. 137–154, 2011.
[12] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1188–1196.
[13] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631. Citeseer, 2013, p. 1642.
[14] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.
[15] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," in *International Conference on Machine Learning*, 2016, pp. 1378–1387.