Lightweight Text Classifier using Sinusoidal Positional Encoding

Byoung-Doo Oh and Yu-Seop Kim
Department of Convergence Software, Hallym University, Republic of Korea
iambd822@gmail.com, yskim01@hallym.ac.kr

Abstract

Large and complex models have recently been developed that require many parameters and much time to solve various problems in natural language processing. This paper explores an efficient way to avoid models being too complicated and ensure nearly equal performance to models showing the state-of-the-art. We propose a single convolutional neural network (CNN) using the sinusoidal positional encoding (SPE) in text classification. The SPE provides useful position information of a word and can construct a more efficient model architecture than before in a CNN-based approach. Our model can significantly reduce the parameter size (at least 67%) and training time (up to 85%) while maintaining similar performance to the CNN-based approach on multiple benchmark datasets.

1 Introduction

In recent years, convolutional neural networks (CNN) have shown remarkable performance and time-efficiency in text classification tasks such as sentiment analysis and document classification. (Kim, 2014; Kalchbrenner et al., 2014; Zhang et al., 2015). In particular, Kim (2014) shows the importance of pre-trained word vectors and fine-tuned word vectors for each task, currently being used in many studies. However, in CNN, convolution and pooling operations lose information about the local order of words (Britz, 2015; Yenigalla et al., 2018). To solve this problem, various studies have been conducted on model architectures capable of effectively extracting features from CNN (Kim, 2014; Kalchbrenner et al., 2014; Zhang et al., 2015; Lai et al., 2015; Zhao et al., 2018; Kim et al., 2020).

On the other hand, when using a word vector based on a distributed representation, fine-tuning of the pre-trained word vector for each task requires more parameters in proportion to the sequence length. Along with this, the proposed large and complex model architectures have improved performance, but additional space and time costs are required due to numerous parameters (Alom et al., 2019). Recently, studies have been conducted on efficient model architectures with less parameter size and computational cost without significant performance loss compared to existing models (Vaswani et al., 2017; Tay et al., 2019; Zhang and Sennrich, 2019).

Vaswani et al. (2017) introduced sinusoidal positional encoding (SPE) and multi-head self-attention to introduce a simple model architecture with less parameter size and computational cost than before. The SPE provides useful word position information without recurrent neural networks (RNN) and, at the same time, requires less parameter size. Due to these advantages, studies have been conducted to show the usefulness of location encoding in computer vision and natural language processing (Takase and Okazaki, 2019; Islam et al., 2020).

In this paper, we propose a single CNN with SPE and construct a simpler model architecture than before with useful position information. SPE's position information is applied to the pre-trained word vector, and it is maintained as a static vector without fine-tuning. A single CNN extracts significant features from word vectors containing position information. The results of this study are confirmed against six benchmark datasets. The proposed model is challenging to achieve state-of-the-art, but it maintained similar performance as before, despite significantly reducing the parameter size and time cost compared to the previous CNN-based approaches.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 63–69

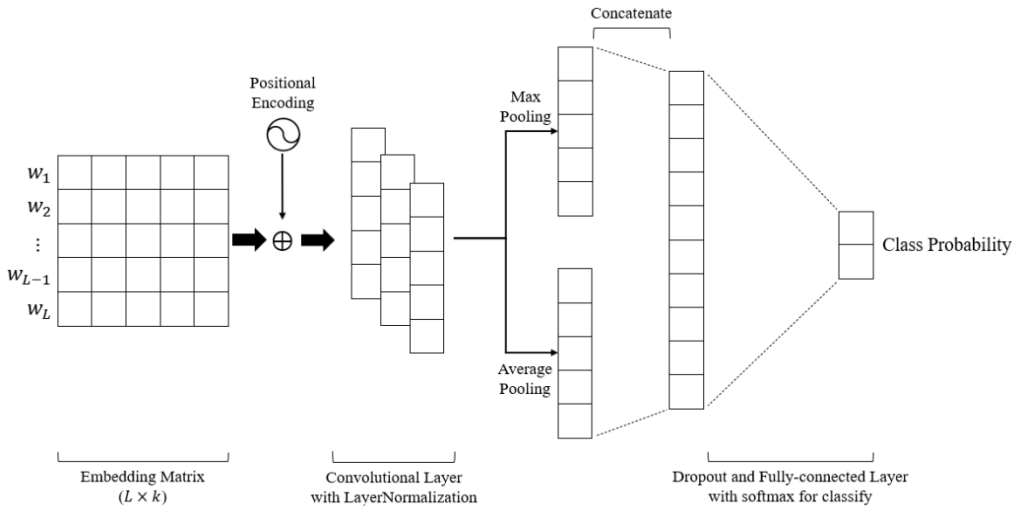

Figure 1: Model architecture of Single CNN with Sinusoidal Positional Encoding.

This paper is organized as follows. Section 2 describes the model proposed in this paper. Section 3 describes hyper-parameters and experimental environments, and Section 4 describes the evaluation and the experimental results. Finally, Section 5 concludes with a summary of what we have identified.

2 Model

As shown in Figure 1, the proposed model's architecture is based on Collobert et al. (2011). The proposed structure is slightly modified. In this paper, a model consisting of SPE and a single CNN is proposed. A single CNN consists of three layers: convolutional layer, pooling layer (max and average), and fully-connected layer with softmax. Each component is described in detail in the rest of this section.

2.1 Sinusoidal Positional Encoding

CNN is difficult to learn word order in sentences (Britz, 2015; Yenigalla et al., 2018). For example, CNN learns "The wolves ate" and "ate the wolves" as the same representation. Therefore, studies have been conducted to effectively provide sequential information to neural network models, excluding RNN (Yang et al., 2016; Gehring et al., 2017; Vaswani et al., 2017).

The SPE introduced by Vaswani et al. (2017) uses sine and cosine functions to represent each word's relative position in an embedding. Besides, it provides useful position information with a parameter-free position representation. SPE ($PE_{(pos,i)}$) is calculated as follows:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/dim}}\right) \quad (1)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/dim}}\right) \quad (2)$$

, where pos is the position of each word in the embedding, i is the position of the dimension in the word vector, and dim is the size of the word vector. Therefore, SPE provides the position information by calculating the word vector's even and odd dimensions with sine and cosine functions, respectively.

In this paper, each word's position information obtained from SPE ($\vec{p}_t$) was added to the word vector (x_i), as shown in Equation (3).

$$\vec{x}_i = x_i + \vec{p}_t \quad (3)$$

The SPE's position information provides useful information when a single CNN extracts features from a word vector. Through this, fine-tuning for word vectors is not considered in training. This can reduce the parameter size required for fine-tuning. The word vector generated in this way is transferred to a single CNN.

2.2 Single Convolutional Neural Networks

$x_i \in \mathbb{R}^k$ is a k-dimensional word vector constituting a sentence, and if the length of the sentence is L, the embedding matrix is represented as $x_{i:L} \in \mathbb{R}^{L \times k}$. The weight of the convolution filter applied to the embedding matrix is represented as $w \in \mathbb{R}^{j \times k}$, and a new feature c_i is generated from j word vectors represented in k-dimensions. For example, feature c_i (when stride=1) is created as follows:

Dataset	c	D_{train}	D_{val}	D_{test}	$\|V\|$	$\|V_{pre}\|$
IMDB	2	22,500	2,500	25,000	112,540	58,843
MR	2	8,635	960	1,067	18,764	16,448
MPQA	2	8,587	955	1,067	6,246	6,083
TREC	6	4,843	539	500	8,689	7,461
Reuters	10	6,472	720	2,787	28,482	17,508
20news	20	10,182	1,132	7,532	117,925	50,021

Table 1: Summary for the datasets after tokenization. c: Number of class. D_{train}: train size. D_{val}: validation size. D_{test}: test size. $|V|$: Vocabulary size. $|V_{pre}|$: Number of words presented in the set of pre-trained word vectors.

$$c_i = f(w \cdot x_{i:i+j-1} + b) \quad (4)$$

In Equation (4), f is the activation function, and $b \in \mathbb{R}$ is the bias value. This convolution filter is applied to all possible j words in a sentence, and all features are concatenated to generate a single feature map c.

$$c = [c_1, c_2, c_3, \dots, c_{i-j+1}] \quad (5)$$

In this paper, the feature map c obtained from the convolutional layer is transferred after normalization. The normalized feature map c applies a max-pooling layer ($\hat{c} = max(c)$) that extracts the maximum value and an average-pooling layer ($\tilde{c} = avg(c)$) that extracts the average value. The obtained features are concatenated and transferred to a fully-connected layer with softmax after regularization to classify the class.

2.3 Normalization and Regularization

We used LayerNormalization (Ba et al., 2016) for feature map c. LayerNormalization is applied independently to each feature map for normalization. Furthermore, since it is normalized to the mean (μ) and variance (σ), it can be applied equally to training and test data and has high time-efficiency.

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} {a_i}^l \quad (6)$$

$$\sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}({a_i}^l - {\mu_i}^l)^2} \quad (7)$$

, where H is the number of hidden nodes in the layer and a_i is the i-th vector at the hidden nodes.

For regularization, we used Dropout (Srivastava et al., 2014) for feature $z = [\hat{c}_1, \dots, \hat{c}_m, \tilde{c}_1, \dots, \tilde{c}_m]$ obtained from the max-pooling layer and the average-pooling layer.

3 Experimental Setup

3.1 Datasets

In this paper, six benchmark datasets - Sentiment classification: Internet Movie Database (IMDB)[1], movie review dataset (MR)[2] (Pang and Lee, 2005). Question categorization: MPQA dataset[3] (Wiebe et al., 2005), TREC question dataset[4] (Li and Roth, 2002). News categorization: Reuters dataset[5] and 20news dataset[6] - are used. Table 1 summarizes each dataset.

3.2 Implementation Details

In this paper, the GloVe[7] (Pennington et al., 2014) vector, which pre-trained with 840 billion words provided by Stanford University, is used, representing the word vector as 300 dimensions.

The hyper-parameters are set the same for all datasets. The activate function used in the convolution layer is a linear function, not a nonlinear function such as ReLU (Nair and Hinton, 2010) or hyperbolic tangent. The filter window size (j) is 3, the number of filters is 128, the l2_regularizer is 0.0001, the epsilon in LayerNormalization is 1e-6, the dropout rate is 0.1, and the mini-batch size is 40. These values were determined experimentally.

Also, we use only early stops when training. As shown in Table 1, the validation dataset is randomly selected, and it is 10% of the training dataset. Furthermore, optimization is performed using Adam optimizer (Kingma and Ba, 2015), and the learning ratio is set to 0.0001.

[1] https://ai.stanford.edu/~amaas/data/sentiment/

[2] https://www.cs.cornell.edu/people/pabo/movie-review-data/

[3] https://mpqa.cs.pitt.edu/

[4] https://cogcomp.seas.upenn.edu/Data/QA/QC/

[5] https://www.nltk.org/book/ch02.html (used only the data of the 10 largest topics)

[6] https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

[7] https://nlp.stanford.edu/projects/glove/

Model	Type	IMDB	MR	MPQA	TREC	Reuters	20news
CNN-static	P	361K	361K	361K	362K	363K	366K
	T	40	**15**	**15**	**8**	12	18
CNN-nonstatic	P	37,637K	6,208K	2,245K	2,969K	9,260K	54,130K
	T	91	18	16	9	15	52
DCNN	P	37,874K	6,445K	2,482K	3,205K	9,497K	54,366K
	T	146	38	36	21	32	77
Capsule-A	P	37,318K	5,889K	1,926K	2,649K	8,941K	53,810K
	T	230	32	30	17	60	111
Capsule-B	P	37,431K	6,002K	2,039K	2,764K	9,058K	53,932K
	T	425	67	65	35	120	213
CapsNet-static	P	47,970K	9,354K	3,158K	7,227K	21,627K	66,503K
	T	287	37	31	21	76	159
CapsNet-Dynamic	P	47,970K	9,354K	3,158K	7,227K	21,627K	66,503K
	T	289	37	31	22	76	160
SingleCNN-static	P	116K	116K	116K	117K	118K	121K
	T	36	15	15	8	10	17
SingleCNN-nonstatic	P	37,392K	5,963K	53,884K	9,015K	2,000K	2,724K
	T	85	18	16	9	13	50
Single CNN-SPE (Proposed Model)	P	**116K**	**116K**	**116K**	**117K**	**118K**	**121K**
	T	**36**	**15**	**15**	**8**	**10**	**17**

Table 2: Experimental results of parameter size and training times (P: parameter size, T: seconds/epoch). The best values in each dataset is shown in bolded.

Model	IMDB	MR	MPQA	TREC	Reuters	20news
CNN-static	89.72	79.38	87.73	90.84	86.83	83.18
CNN-nonstatic	89.69	**81.09**	88.42	91.12	87.17	**84.63**
DCNN	89.6	79.53	89.26	89.32	87.06	81.55
Capsule-A	86.16	78.22	86.93	82.94	82.29	60.65
Capsule-B	89	79.74	88.19	82.95	**88.1**	69.97
CapsNet-static	87.18	77.96	88.75	92.08	87.58	81.34
CapsNet-Dynamic	86.47	78.31	**89.2**	92.35	87.45	82.42
SingleCNN-static	90.37	80.23	88.43	91.08	86.03	81.71
SingleCNN-nonstatic	**90.58**	80.02	88.41	91.13	86.91	83.78
SingleCNN-SPE (Proposed Model)	90.44	80.93	88.88	**92.52**	86.89	82.95

Table 3: Experimental results of accuracy. Models marked with an asterisk used the code published by the author on GitHub for accurate implementation.

3.3 Baseline Model

In this paper, to confirm our model's performance, we compare it with various models using only CNN as follows: CNN-static and CNN-nonstatic (Kim, 2014), DCNN (Kalchbrenner et al., 2014), Capsule-A and Capsule-B (Zhao et al., 2018), CapsNet-static and CapsNet-dynamic (Kim et al., 2020). These models are set identically to the hyper-parameters proposed in each paper.

We also identify a model that does not use SPE to confirm SPE's usefulness in the CNN-based approach and are as follows: SingleSCNN-static (without fine-tuning) and SingleSCNN-nonstatic (with fine-tuning).

4 Results and Discussion

First, we check the parameter size and training time. When measuring the training time, all models set the mini-batch size to 5. The results are shown in Table 2.

Each model, except the proposed model, performs fine-tuning in training. The size of the parameters, including this, is shown in Table 2. In this case, our model reduces the parameter size significantly, which is compared to the previous one. Even in excluding fine-tuning, the parameter size could be reduced by about 67% (CNN-static).

The training times are reduced by up to 90% based on the IMDB with the large-scale dataset,

Model	MR	MPQA	TREC	Reuters
SingleCNN-static	71.68	82.86	83.76	84.91
SingleCNN-SPE	52.26	67.77	86.6	83.76

Table 4: Experimental results using simple bag-of-words.

and only CNN-static shows a similar training time to our model.

Table 3 shows the performance of the models composed of the parameter size in Table 2. In the performance evaluation, all methods are repeated five times, and the average accuracy is measured. Our model maintains similar performance to other models, despite reducing the parameter size by at least 67%.

Additionally, we confirmed SPE's usefulness for word vectors represented based on simple bag-of-words, as shown in Table 4. Originally, SPE calculates each word's position information from the word vector based on the distributed representation. Therefore, as shown in Table 3, when the word vectors based on distributed representation was used, SPE's usefulness was confirmed. Experimental results with bag-of-words based word vectors were not good. It was also confirmed that the SPE's position information is difficult to provide efficient information in the bag-of-words based word vector.

4.1 SPE vs Fine-tuning

Kim (2014) confirmed the usefulness of fine-tuning for pre-trained word vectors, and it is still used in many research. However, this process requires additional parameters in addition to the parameters of the model in training. Therefore, we compared the performance with SingleCNN-static and SingleCNN-nonstatic to confirm SPE's usefulness in the CNN-based approach.

As shown in Table 3, the position information obtained by the SPE confirmed a result similar to the effect of fine-tuning in the CNN-based approach. It is considered that the position information obtained by the SPE can have an efficient effect without fine-tuning in the CNN-based approach.

Additionally, other models' performance was different from that of the previous paper, which is considered to have arisen from the experimental environment's difference.

5 Conclusion

In this paper, we propose a single convolutional neural network to which sinusoidal positional encoding is applied. Besides, we describe an experiment to confirm the usefulness of sinusoidal positional encoding in the CNN-based approach. This model shows excellent performance despite being lightweight in text classification. Although this model reduced the parameter size by at least 67%, it was possible to confirm a similar performance to the previous model. Besides, a similar effect could be confirmed without fine-tuning in the CNN-based approach.

In the future, we intend to study a method that can easily learn the position information of words in a manner other than SPE. Through this, we want to develop a more straightforward and more powerful model that can show both the time-advantageous CNN and the sequential learning power of RNN.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C2006010).

Reference

Md Zahangir Alom, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C. Van Essen, Abdul A. S. Awwal, and Vijayan K. Asari. 2019. A state-of-the-art survey on deep learning theory and architectures. Electronics, 8(3):292. https://doi.org/10.3390/electronics8030292.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. arXiv preprint arXiv:1607.06450.

Denny Britz. 2015. Understanding convolutional neural networks for NLP. http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from

Scratch. Journal of machine learning research, 12:2493-2537.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In Proceedings of the 34th International Conference on Machine Learning (Volume 70), pages 1243-1252.

Md Amirul Islam, Sen Jia, and Neil D. B. Bruce. 2020. How Much Position Information Do Convolutional Neural Networks Encode?. In Proceedings of the 8th International Conference on Learning Representations.

Nal Kalchbrenner, Edward Grefenstette, and Phill Blunsom. 2014. A convolutional neural network for modelling sentences. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 655-665. https://www.aclweb.org/anthology/P14-1062.

Jaeyoung Kim, Sion Jang, Eunjeong Park, and Sungchul Choi. 2020. Text classification using capsules. Neurocomputing, 376:214-221. https://doi.org/10.1016/j.neucom.2019.10.033.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 1746-1751. https://www.aclweb.org/anthology/D14-1181.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. In Proceedings of the Twenty-ninth AAAI Conference on Artificial Intelligence, pages 2267-2273.

Xin Li and Dan Roth. 2002. Learning Question Classifiers. In Proceeding of the 19th International Conference on Computational Linguistics. https://www.aclweb.org/anthology/C02-1150.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on Machine Learning, pages 807-814.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pages 115-124. https://www.aclweb.org/anthology/P05-1015.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 1532-1543. https://www.aclweb.org/anthology/D14-1162.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of machine learning research, 15(1):1929-1958.

Sho Takase and Naoaki Okazaki. 2019. Positional Encoding to Control Output Sequence Length. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3999-4004. https://www.aclweb.org/anthology/N19-1401.

Yi Tay, Aston Zhang, Luu Anh Tuan, Jinfeng Rao, Shuai Zhang, Shuohang Wang, Jie Fu, and Siu Cheung Hui. 2019. Lightweight and Efficient Neural Natural Language Processing with Quaternion Networks. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1494-1503. https://www.aclweb.org/anthology/P19-1145.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In Advances in Neural Information Processing Systems, pages 5998-6008.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating Expressions of Opinions and Emotions in Language. Language Resources and Evaluation, 39(2-3):165-210. https://doi.org/10.1007/s10579-005-7880-9.

Yunlun Yang, Yunhai Tong, Shulei Ma, and Zhi-Hong Deng. 2016. A Position Encoding Convolutional Neural Network Based on Dependency Tree for Relation Classification. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 65-74. https://www.aclweb.org/anthology/D16-1007.

Promod Yenigalla, Sibsambhu Kar, Chirag Singh, Ajay Nagar, and Gaurav Mathur. 2018. Addressing Unseen Word Problem in Text Classification. In Proceedings of the International Conference on Applications of Natural Language to Information Systems, pages 339-351. https://doi.org/10.1007/978-3-319-91947-8_36.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In Advances in Neural Information Processing Systems, pages 649-657.

Biao Zhang and Rico Sennrich. 2019. A Lightweight Recurrent Networks for Sequence Modeling. In

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1538-1548. https://www.aclweb.org/anthology/P19-1149.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Soufei Zhang, and Zhou Zhao. 2018. Investigating Capsule Networks with Dynamic Routing for Text Classification. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3110-3119. https://www.aclweb.org/anthology/D18-1350.