A Neural Language Model with Attention Mechanism Based on Convolutional Neural Network

Seongik Park, Ki Yong Lee, and Yanggon Kim

Abstract—**The convergence of artificial intelligence (AI) technology and natural language processing (NLP) has rapidly increased the demands for an analysis on the natural language that involves plenty of ambiguities not present in formal language. For this reason, the language model (LM), a statistical approach, has been used as a key role in this area. Recently, the emerging field of deep learning, which applies complex deep neural networks for machine learning tasks, has been applied to language modeling and achieved more remarkable results than traditional language models. One of the important techniques that have led neural network-based LM success is the attention mechanism. Attention mechanism makes neural networks pay attention to specific words in the input sentence when generating the output words. However, although the attention mechanism has improved the performance of many neural network models, it requires tons of parameters to achieve the state-of-art level performance. This is because attention mechanism encodes the context of a word by simply accumulating the outputs from the network for all the input words, which may cause information loss. To compensate for this limitation, we propose an extension of attention mechanism by adopting a convolutional neural network to replace the accumulation. With only far fewer parameters, our model achieved comparable performance to the recent state-of-the-art models on the very popular benchmark datasets, yielding perplexity scores of 58.4 on the Penn Treebank dataset and 50.1 on the Wikitext-2 dataset, respectively.**

Index Terms—**Attention mechanism, deep learning, neural language model, neural network.**

I. Introduction

In the field of natural language processing (NLP), the process of language modeling (LM) seeks to build a statistical model that approximates the probability distribution of subsequent words of a natural language given the preceding terms [1]. Given a sentence S, the probability of S can be calculated as $P(S) = P(w_1, w_2, \dots, w_n)$, where w_1, w_2, …, w_n are words in S and n is the length of S [2]. The emerging field of deep learning, which applies complex deep neural networks for machine learning tasks, has been adopted to improve all aspects of statistical data analysis, including in NLP. A particular type of recurrent neural network (RNN) proven to be robust in processing sequential data is the Long-Short Term Memory (LSTM) [3] network. A variety of LSTM-based language models have been proposed and achieved better performance than other RNNs and traditional language models [4]-[7].

Attention mechanism is one of the most important techniques that has improved the performance of neural network-based NLP. Many studies have applied attention mechanism to their neural networks and have shown state-of-the-art results in various NLP tasks such as machine translation and language model [8], [9]. Attention mechanism encodes the relevant context information of a given word from distant parts of a sentence by calculating a context vector which is a weighted sum of the outputs from the network and passes the context vector to the decoder to generate a word. In language modeling, specifically, attention helps to find a relationship between the target word which will be generated and the preceding words (or context) in a sentence by analyzing which words have a strong relationship with the target word. However, attention mechanism calculates a context vector by simply accumulating the weighted sum of the outputs from the network, which may cause information loss. As a result, many attention-based models tend to use tons of parameters to train their models and to achieve the state-of-the-art level performance. The size of neural networks is very important because the smaller the neural network, the smaller the memory requirement, power consumption, and the time required for the training [10], [11].

In this paper, therefore, we propose a neural network-based language model with an extended attention mechanism. We customized attention mechanism by minimizing information loss that occurs while encoding the source context. Instead of simple accumulation of the outputs from the network into a context vector, our model makes a stack of the outputs from the network and then applies a convolutional neural network (ConvNet) with a 1x1 filter, which can reduce the dimension of the stack for faster computation and achieve less information loss [12]. Through this minimizing information loss capability of the customized attention mechanism, our model achieves comparable performance to the recent state-of-the-art models with only far fewer parameters.

II. Related Work

A. Neural Network-Based Language Model

Neural network-based language models have achieved

Manuscript received August 25, 2020; revised March 11, 2021. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00269, a research on safe and convenient big data processing methods).

Seongik Park and Yanggon Kim are with the Department of Computer & Information Sciences at Towson University, USA (e-mail: spark32@students.towson.edu, ykim@towson.edu).

Ki Yong Lee is with the Division of Computer Science at Sookmyung Women's University, South Korea (e-mail: kiyonglee@sookmyung.ac.kr).

remarkable results compared to traditional language models. One of the most important techniques which have led to the success of applying a neural network to the LM is the recurrent neural network (RNN). RNN uses its internal memory (or cell) to process input sequences of arbitrary length. As a result, RNN can be applied to many NLP-based tasks such as speech recognition and language modeling. Currently, Long-Short Term Memory (LSTM) [3] has frequently been used instead of the traditional RNN. Using LSTM allows the network to continue learning over many time steps by learning long-term dependencies using internal cell states. Many LSTM-based language models [4]-[7] have shown improvement over RNN-based language models.

LSTM's cell contains two states named hidden state at time step *t*, h_t, and the cell state at time step *t*, C_t. h_t and C_t memorize preceding information and carry it toward the cascading cells. However, it is hard to memorize all the information of preceding cells because each of them is represented by only one matrix. Consequently, this can cause a vanishing memory problem. This has led attention mechanism being proposed and applied to RNN-based neural networks for many NLP tasks.

B. Attention Mechanism

Attention is one of the most influential ideas in deep learning. The purpose of the attention is to make a model pay attention to the important areas such as contexts in NLP and objects in computer vision tasks. Fig. 1 shows an example of the attention mechanism applied on an object detection task. The attention mechanism in this example helps the model focus on the areas that contain important information for object detection.

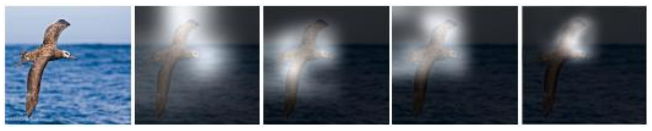
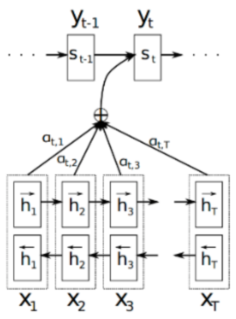


Fig. 1. An example of attention mechanism for an object detection task in Zhao, Bo, *et al*. [13].


y_{t-1}
y_t
s_{t-1}
s_t
$\alpha_{t,1}$
$\alpha_{t,2}$
$\alpha_{t,3}$
$\alpha_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
x_1 x_2 x_3 x_T
Fig. 2. An example of attention mechanism for a NLP task in Bahdanau *et al.* [14].

The attention mechanism in NLP was originally proposed in the "encoder-decoder" architecture for neural machine translation [14]. The encoder-decoder model encodes the input sequence to one fixed length vector and decodes the vector to generate the output sentence. Attention in the encoder-decoder model helps to find which parts of the input words are relevant to each word in the output sentence from the encoded vector, allowing modeling of dependencies regardless of their distance in the input or output sentences. As shown in Fig. 2, given input words x, the encoder-decoder model generates its output y from hidden state s which is calculated as $s_i = f(s_{i-1}, y_{i-1}, c_i)$. The context vector c for the current output word is calculated as $c_i = \sum_{j=1}^{T_x} h_j \alpha_{ij}$ where T is the number of words in the input sentence, a_{ij} is an attention weight and h_j is a sequence of annotations to which an encoder maps the input sentence [14]. This context vector summarizes the meaning of the whole input sentence for the current output word.

The attention mechanism has been applied to many different NLP-based approaches. Self-attention [8] was introduced for machine translation task. A self-attention module computes the relationship at a position in a sequence by paying attention to all positions and taking their weighted average in an embedding space. In addition, BERT [9], a deep learning model that has given state-of-the-art results on a wide variety of natural language processing tasks, also applied self-attention mechanism to its model. Even though they showed remarkable performances, the problem of information loss is still remaining because of its simple accumulation process for the calculation of the context vector *c* described earlier. As a result, many attention-based models tend to use tons of parameters to achieve the state-of-the-art level performance.

III. System Architecture

A. Proposed Language Model

Fig. 3 shows the architecture of the proposed language model. ConvNet in the figure represents a convolutional neural network that performs a convolution operation with a 1×1 filter stride by 1 at a time. Batch norm and fc denote a batch normalization layer and a fully-connected layer, respectively. Finally, ⊕ is the element-wise matrix addition operator.

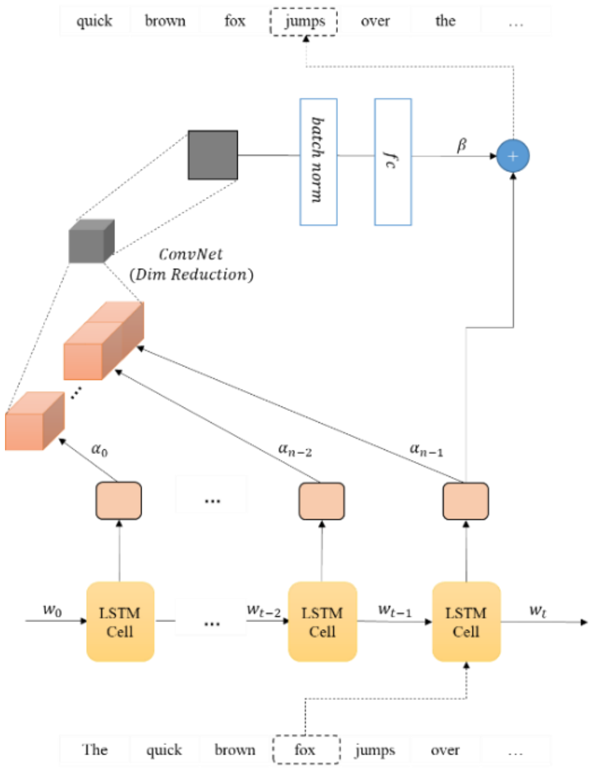


Fig. 3. Proposed architecture with an example sentence.

Our proposed model employs the basic idea of the attention mechanism described in Section II.B. That is, when predicting the next possible word given its preceding words in a sentence, it focuses and decides which preceding words have a strong relationship with the predicting word by adopting the attention mechanism. Basically, the attention mechanism accumulates feature maps (i.e., hidden states) into a 2-D matrix which is called a context vector. In contrast, our model *stacks* feature maps into a 3-D matrix. However, using a 3-D matrix as a parameter for further calculation causes a dimensional inequality problem since its dimension is different from that of the feature map (i.e., 2-dimensions), which resulted from the last inputted word. Therefore, a *ConvNet* with a 1×1 filter is applied to reduce the dimension of the stack so that the dimension of the stack is reduced to that of the feature map. The implication of dimensionality reduction is to exclude unimportant information from the preceding words, which is similar to what the attention mechanism does. After applying the *ConvNet*, the batch normalization layer (*batch norm*) normalizes the dimension-reduced preceding information to decrease the dependency of weights initialization and to prevent the overfitting problem. A fully-connected layer (*fc*) is applied to match the shape of the result with the last feature map. Finally, the result from our customized attention is added to the last feature map.

B. Layers in Proposed Model

1) Embedding & LSTM layer

Because the neural network only takes numerical values as its input, word embedding, a technique to convert textual data to vector representation [2], [6], [7], [15], [16], is required. We employ the word embedding layer at the very first position of our model to transform words into vectors. Let V refer to the word vocabulary that stores every presented word in our dataset. Given a word w_t at time step t, w_t is converted into a vector and transformed into matrix x_t with size $n \times k$, where n is the size of V and k is the dimensionality of embedding.

LSTM solves the problem of learning long dependencies by using memory cells at each time step. LSTM mainly contains three gates, i.e., input gate I, forget gate f, and output gate o at each cell. At time step t, LSTM takes an embedded word as input x_t and calculates the hidden state h_t and cell state C_t via the following:

$$\begin{gathered} i_t = \sigma(W_i \odot [h_{t-1}, x_t] + b_i), \\ f_t = \sigma(W_f \odot [h_{t-1}, x_t] + b_f), \\ o_t = \sigma(W_o \odot [h_{t-1}, x_t] + b_o), \\ g_t = tanh(W_c \odot [h_{t-1}, x_t] + b_C), \\ C_t = f_t \odot C_{t-1} + i_t \odot g_t, \\ h_t = o_t \odot tanh(C_t), \end{gathered} \tag{1}$$

where W_i, W_f, W_o, and W_c are the weights, b_i, b_f, b_o, and b_C are the biases, h_{t-1} is the old hidden state, and C_{t-1} is the old cell state. Operator $\odot$ is the element-wise matrix multiplication, and $\sigma(*)$ and $tanh(*)$ are non-linearity functions that denote the element-wise sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ and hyperbolic-tangent $tanh(x) = \frac{2}{1+e^{-2x}} - 1$, respectively.

2) Convolutional attention layer

The attention mechanism originally computes a weighted sum of feature maps, where each feature map h_t has a weight a_t, and accumulates them into a context vector c. Here, $0 \le a_t \le 1$ and their total sum is 1. Compared to this, the proposed approach makes a stack of feature maps. Each feature map h_t at time step t is multiplied with α_t and stacked into a 3-D matrix called the context map c_m as:

$$c_m = stack(h_1\alpha_1, h_2\alpha_2, \dots, h_T\alpha_T). \tag{2}$$

After that, to deal with the inequality of dimensions, a convolutional neural network with a 1×1 filter [17], [18] is applied to reduce the dimension of the context map c_m. This convolutional neural network with a 1×1 filter is well known for faster computation and less information loss [12]. With context map c_m, a convolutional neural network performs a convolutional operation with a 1×1 filter stride by 1 at a time. The padding is unnecessary, because the convolutional neural network has the exact same output size as the input, unlike a 3×3 filter, which is the most common filter size for convolutional neural networks. The output size w_o can be calculated by the following:

$$w_o = \frac{w_i - f + 2p}{s} + 1, \tag{3}$$

where w_i is the size of the width of an input, f is the size of the filter, p is the size of the padding, and s is the size of the stride. According to (3), no matter what the size of input w_i is, the size of output w_o is always exactly the same as w_i, since f is 1, p is 0 and s is 1. In addition, the purpose of this convolution is the dimensionality reduction, thus a pooling layer is also unnecessary. In order to compute the dimension-reduced context map, i.e., the new context vector c obtained from the context map c_m, the convolution layer sums up the contributions:

$$c = \sum_{x=0}^{rows} \sum_{y=0}^{columns} \sum_{z=0}^{depth} w_{x,y,z} \cdot c_{m(i-x,j-y,k-z)}. \tag{4}$$

Here, $c_{m(i,j,k)}$ represents the element at the ith row, the jth column, and the kth depth in c_m. This convolutional layer with a 1x1 filter reduces the dimensionality of a context map c_m into c. In addition, the 1x1 filter is trained by the network so that the network can decide which information can remain, replacing the simple accumulation process of the traditional attention mechanism. The output of this convolutional layer is passed to the following batch normalization (*batch norm*) and fully-connected layer (*fc*).

Batch normalization is one of the ideas to prevent the gradient vanishing and the gradient exploding problems by normalizing layer inputs to reduce internal covariate shift [19]. We adopt *batch norm* at the end of the convolutional layer to accelerate the training rate and to prevent overfitting. *fc* is used to match the shape of c to h_t, and β is a trainable weight which is used to calculate an attention vector from the result of *fc*. We recommend to initialize both *fc* and β with

Xavier Initialization [20] which initializes the weights from a distribution with zero mean and variance $2/(n_{in} + n_{out})$ where n_{in} and n_{out} are respectively the number of inputs and outputs of the layer, otherwise the gradient of the network tends to exploding.

Afterwards, an attention vector is calculated by multiplying the result from fc and β. Finally, the attention vector is used to calculate the final result by adding to h_t. Once the new h_t is calculated, the softmax function, which converts the vector of arbitrary real values to another vector in which each element is in the range 0 to 1 where the total sum is 1, calculates the output to obtain probability distribution over the next word.

IV. Experiment

A. Dataset

The Penn Treebank (PTB) dataset [21] is one of the most popular benchmark datasets to measure the quality and performance of the language models. This PTB dataset was preprocessed by Mikolov *et al*. [15]. They replaced rare words such as names of persons and companies with a special symbol '<unk>' and numerical words such as age and date with a symbol 'N', which are hard to be predicted by language models because of its uniqueness. The Wikitext-2 dataset [22] is also a very popular benchmark dataset. Compared to PTB dataset, Wikitext-2 dataset, which is extracted from Wikipedia articles, has been proposed as a more realistic benchmark containing numbers, cases, and punctuations.

TABLE I: The Penn Treebank and Wikitext-2 Dataset

	PTB			**Wikitext-2**		
	Train	Valid	Test	Train	Valid	Test
Articles	2,000	155	155	600	60	60
Tokens	930K	74K	82K	2M	211K	238K
# of Vocab*		10,000			33,231	
OoV**		4.8%			2.6%	

*# of Vocab: Number of words in the vocabulary
**OoV: Out-of-Vocabulary

As shown in Table I, the preprocessed PTB dataset is composed of 930K words in the training set, 74K words in the validation set, and 82K words in the test set, respectively. Wikitext-2 dataset consists of 2M tokens for training set, 211K for validation, and 238K tokens for test set. We conducted this experiment on both preprocessed PTB dataset and Wikitext-2 dataset.

B. Evaluation and Discussion

1) Evaluation and comparison

Perplexity (*PPL*) is a well-known measurement for language modeling, indicating how well a probability model predicts a sample. *PPL* is the exponential of the average negative log-likelihood. Given words w_1, w_2, …, w_m in a sentence *S*, *PPL* is calculated as:

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m} \log_e P(w_i),$$
$$PPL = e^{\mathcal{L}}, \tag{5}$$

where m is the number of words in S and $P(w_i)$ is the likelihood of w_i. A lower *PPL* indicates better performance. In this experiment, we evaluated our model based on *PPL* score.

We first evaluated our approach and compared with the traditional LSTM model. Table II shows the performance of models based on *PPL* score. As shown in Table II, *PPL* scores for the test set have been improved from 115.0 to 58.4 for PTB dataset and from 143.0 to 50.1 for Wikitext-2 dataset. There is about 50% and 65% of improvement, respectively. This result shows that applying our approach to the traditional LSTM based language model is able to improve the performance in terms of PPL with only a small amount of network size increased.

TABLE II: Comparison of Performance Based on *PPL* Score

	PTB			**Wikitext-2**		
	LSTM	Ours	Imp.*	LSTM	Ours	Imp.
Valid	119.2	61.3	48.55%	150.2	53.7	64.25%
Test	115.0	58.4	49.17%	143.0	50.1	64.97%

*Imp.: Improved

Table III shows the comparison of performance with various recent studies based on *PPL* score. *FRAGE + AWD-LSTM-MoS + dynamic eval* in Table III is referred to as the state-of-the-art performance for both PTB and Wikitext-2 dataset. Even though our proposed approach does not achieve the state-of-the-art performance, it shows reasonable *PPL* score considering the number of parameters required to train the model. For PTB dataset, our model requires only 8M number of parameters to achieve 58.46 *PPL* score. The state-of-the-art *PPL* is 46.54, and that model requires 22M number of parameters. Similarly, with the same amount of parameters as used for PTB dataset, the proposed model shows 50.1 *PPL* score for Wikitext-2 dataset. Compared to the state-of-the-art model which requires 35M for the training, our model requires a much smaller number of parameters yet achieves relatively reasonable performance.

TABLE III: Comparison of Performance with State-of-the-Art Models Based on PPL Score

Models	**PTB**			**Wikitext-2**		
	Valid	Test	# of Param*	Valid	Test	# of Param
Variational RHN (Zilly *et al*. 2016)	82.62	78.29	32M	-	-	-
Transformer-XL (Zihang *et al.* 2019)	56.72	54.52	24M	-	-	-
AWD-LSTM-DOC (Takase, S. *et al.* 2018)	54.12	52.38	23M	60.29	58.03	37M
AWD-LSTM-DOC + Partial Shuffle (Ofir Press, 2019)	53.79	52.0	23M	60.16	57.85	37M
FRAGE + AWD-LSTM-MoS + dynamic eval (Gong, C. *et al.* 2018)	**47.38**	**46.54**	22M	**40.85**	**39.14**	35M
Ours	61.33	58.46	**8M**	53.7	50.1	**8M**

2) Discussion

The size of neural networks is a very important matter. The smaller the neural network, the smaller the memory requirement, power consumption, and the time required for the training. Han *et al.* [10] mentioned neural networks are both computationally intensive and memory intensive, making them difficult to deploy on embedded systems with limited hardware resources. Schwartz *et al*. [11] also argue that the financial cost of the computations can make it difficult for academics, students, and researchers, in particular those from emerging economies, to engage in deep learning research. They also define and advocate the use of Green AI referring to AI research that yields novel results without increasing computational cost instead of Red AI which seeks to obtain state-of-the-art results in accuracy (or related measures) through the use of massive computational power. These researches show that our proposed model may outperform other state-of-the-art models in terms of resource efficiency.

V. Conclusion

We proposed a neural language model based on LSTM network with an extended attention mechanism to mitigate information loss resulting from the simple accumulation process of the traditional attention mechanism. Our model makes a stack of the outputs from the LSTM network and then applies a convolutional neural network with a 1x1 filter, which can reduce the dimension of the stack for faster computation and less information loss. After that, the newly calculated context vector goes through a batch normalization layer and fully-connected layer to minimize the dependency of weights initialization and to prevent the overfitting problem. We evaluated the performance of our model on the very popular benchmark datasets which are PTB and Wikitext-2 datasets and compared them with the traditional LSTM-based language model and recent studies. Our model improved PPL scores of the traditional LSTM model from 115.0 to 58.4 on PTB and from 143.0 to 50.1 on Wikitext-2 dataset, respectively. In addition, the state-of-the-art model for both PTB and Wikitext-2 datasets requires 22M and 35M number of parameters to achieve the performances. Compared to this, our model only requires 8M number of parameters yet shows relatively reasonable performance.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Seongik Park and Yanggon Kim conducted the research; Seongik Park as the first author analyzed the data and wrote the paper; Ki Yong Lee and Yanggon Kim provided suggestions for the research and revised the paper; all authors had approved the final version.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of Machine Learning Research*, vol. 3, pp. 1137-1155, Feb. 2003.

[2] C. Trim. (April 26, 2013). *What is Language Modeling?* [Online]. Available: http://trimc-nlp.blogspot.com/2013/04/language-modeling.html

[3] H. Sepp and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[4] M. Sundermeyer, R. Schlüter, and H. Ney, “LSTM neural networks for language modeling,” in *Proc. Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Advances in Neural Information Processing Systems*, pp. 3104-3112, 2014.

[6] W. Zaremba, I. Sutskever, and O. Vinyals, “Recurrent neural network regularization,” arXiv preprint arXiv: 1409.2329, 2014.

[7] J. G. Zilly, R. K. Srivastava, J. Koutník, and J. Schmidhuber, “Recurrent highway networks,” arXiv preprint arXiv: 1607.03474, 2016.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones *et al.*, “Attention is all you need,” *Advances in Neural Information Processing Systems*, 2017, pp. 5998-6008.

[9] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv: 1810.04805, 2018.

[10] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” arXiv preprint arXiv: 1510.00149, 2015.

[11] R. Schwartz, J. Dodge, N. Smith, and O. Etzioni, “Green AI,” arXiv preprint arXiv: 1907.10597, 2019.

[12] C. Szegedy, W. Liu, Y. Jia *et al*., “Going deeper with convolutions,” in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1-9.

[13] B. Zhao, X. Wu, J. Feng, Q. Peng, and S. Yan, “Diversified visual attention networks for fine-grained object classification,” *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1245-1256, 2017.

[14] B. Dzmitry, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv: 1409.0473, 2014.

[15] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, “Recurrent neural network based language model,” in *Proc. Eleventh Annual Conference of the International Speech Communication Association*.

[16] M. Tomas *et al.*, “Distributed representations of words and phrases and their compositionality,” *Advances in Neural Information Processing Systems*, 2013.

[17] S. Park, J. H. Song, and Y. Kim, “A neural language model for multi-dimensional textual data based on CNN-LSTM network,” in *Proc. 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2018, pp. 212-217.

[18] M. Lin, Q. Chen, and S. Yan, “Network in network,” arXiv preprint arXiv: 1312.4400, 2013.

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

[20] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proc. the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249-256.

[21] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, “Building a large annotated corpus of English: The Penn Treebank,” *Computational Linguistics*, vol. 19, no. 2, pp. 313-330, 1993.

[22] S. Merity, C. Xiong, J. Bradbury, and R. Socher, “Pointer sentinel mixture models,” CoRR,abs/1609.07843, 2016.

Seongik Park received his B.S. degree in computer science from National Institute for Lifelong Education, Republic of Korea in 2012 and M.S. and D.Sc. degrees in computer science and information technology from Towson University, Maryland, USA in 2016 and 2020, respectively. His research interests include deep learning, natural language process, machine learning, big data and data mining.

Ki Yong Lee received his B.S., M.S., and Ph.D. degrees in computer science from KAIST, Daejeon, Republic of Korea, in 1998, 2000, and 2006, respectively. From 2006 to 2008, he worked for Samsung Electronics, Suwon, Korea as a senior engineer. From 2008 to 2010, he was a research assistant professor of the Department of Computer Science at KAIST, Daejeon, Korea. He joined the Faculty of the Division of Computer Science at Sookmyung Women's University, Seoul, in 2010, where currently he is a professor. His research interests include database systems, query processing, data mining, data streams, and scientific data processing.

Yanggon Kim received his B.S. and M.S. degree from Seoul National University, Seoul, Korea in 1984 and 1986, respectively and Ph.D. degree in computer science from Pennsylvania State University, Pennsylvania, 1995. He is currently a professor in the Department of Computer and Information Sciences, Towson University. His current research interests include computer network, secure BGP network, distributed computing systems, big data and data mining in social networks.