KIPS Trans. Comp. and Comm. Sys.
Vol.9, No.1 pp.17~24 pISSN: 2287-5891


https://doi.org/10.3745/KTCCS.2020.9.1.17

# CNN Architecture Predicting Movie Rating from Audience's Reviews Written in Korean

Hyungchan Kim† · Heung-Seon Oh†† · Duksu Kim†††

## ABSTRACT

In this paper, we present a movie rating prediction architecture based on a convolutional neural network (CNN). Our prediction architecture extends TextCNN, a popular CNN-based architecture for sentence classification, in three aspects. First, character embeddings are utilized to cover many variants of words since reviews are short and not well-written linguistically. Second, the attention mechanism (i.e., squeeze-and-excitation) is adopted to focus on important features. Third, a scoring function is proposed to convert the output of an activation function to a review score in a certain range (1-10). We evaluated our prediction architecture on a movie review dataset and achieved a low MSE (e.g., 3.3841) compared with an existing method. It showed the superiority of our movie rating prediction architecture.

Keywords : NLP, CNN, Movie Rating, Un-Normalized Text Data

# 한국어 관객 평가기반 영화 평점 예측 CNN 구조

김 형 찬† · 오 흥 선†† · 김 덕 수†††

## 요 약

본 논문에서는 합성곱 신경망 기반의 영화 평점 예측 구조를 제안한다. 제안하는 구조는 문장 분류을 위하 고안된 TextCNN를 세 가지 측면에서 확장하였다. 첫 번째로 문자 임베딩을 이용하여 단어의 다양한 변형들을 처리할 수 있다. 두 번째로 주목 메커니즘을 적용하여 중요한 특징을 더욱 부각하였다. 세 번째로 활성 함수의 출력을 1-10 사이의 평점으로 만드는 점수 함수를 제안하였다. 제안하는 영화 평점 예측 구조를 평가하기 위해서 영화 리뷰 데이터를 이용하여 평가해 본 결과 기존의 방법을 사용했을 때보다 더욱 낮은 MSE를 확인하였다. 이는 제안하는 영화 평점 예측 구조의 우수성을 보여 주었다.

키워드 : 자연어처리, CNN, 영화 평점, 비정제 문자 데이터

## 1. Introduction

Convolution neural network (CNN) is effective and successful for handling visual tasks [1]. Recently, CNN also has employed in non-visual tasks like speech recognition [2], natural language processing (NLP) [3], and so on. Sentence classification is a fundamental operation in NLP, and we can employ a well-designed network in many applications like classifying the class of text, e.g., predicting a rating for a movie from reviewing text. Some recent works have demonstrated that a simple CNN model performs well for sentence classification tasks [4]. However, most of those CNN models were designed for normalized texts written in English.


Fig. 1. Examples of Movie Reviews and Ratings in the NAVER Movie Site

※이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(No.2018R1C1B5045551) 및 2019학년도 한국기술교육대학교 교수 교육연구진흥과제의 지원에 의하여 연구되었음.
† 비 회 원 : 한국기술교육대학교 컴퓨터공학부 학사과정
†† 비 회 원 : 한국기술교육대학교 컴퓨터공학부 조교수
††† 정 회 원 : 한국기술교육대학교 컴퓨터공학부 조교수
Manuscript Received : October 18, 2019
Accepted : November 15, 2019
* Corresponding Author : Duksu Kim(bluekds@koreatech.ac.kr)

**Contributions:** In this paper, we introduce a novel prediction system that translates an audience's review written in Korean for a movie to a rating. Our prediction architecture extends TextCNN [4], a popular CNN-based architecture for sentence classification in three aspects. First, character embedding is utilized to cover a lot of variants of words since reviews are short and not well-written linguistically. 1) sentences (i.e., reviews) are written in Korean and 2) that contain lots of slang (e.g., a set of consonants) and incomplete sentences. Focusing on characters rather than words, we can avoid word-mismatching and reduce the size of the vocabulary (Sec. 3.1). Second, attention mechanism, squeeze-and-excitation (SE), is adopted to focus on important features. In CNN, SE blocks capture spatial correlation among features efficiently (Sec. 3.2). Third, a scoring function is proposed to convert the output of an activation function w.r.t. input review to a review score in a certain range (1-10) (Sec. 3.3).

To evaluate our prediction architecture, a series of experiments were performed based on a review dataset. We constructed our review dataset from NAVER movie (https://movie.naver.com/), which is one of the largest movie rating and ticketing services in South Korea. Fig. 1 shows two examples of movie ratings in NAVER movie. An audience of a movie made a review score between 1 and 10. It indicates a good movie as the score is close to 10. Based on the review dataset, five different methods, two prior methods, and three variations of our method are compared using mean square error (MSE). From the results, our method reduces MSE up to 3.3841. It shows that all variations of our method achieved improvements over the prior methods. As a result, it reveals that our prediction architecture has superiority in movie ratings.

## 2. Related Work

The core of the convolution neural network (CNN) is convolution layers and pooling layers. A convolution layer outputs a feature map that represents the spatial and local connectivity of an input. A pooling layer outputs a representative feature of a feature map [1]. Since CNN reduces the dimensionality of network effectively compared with a fully-connected network while maintaining features of multidimensional data (e.g., image), it has been widely employed, and prior works have proved its usefulness and effectiveness in visual task [1, 5].

**CNN models for sentence classification:** Recently, CNN has also applied for other applications, including sentence classification in NLP. Kim [4] introduced a CNN architecture, TextCNN, for sentence classification. TextCNN uses multiple convolution layers having a different kernel size in a parallel manner. With this shallow-and-wide CNN model, they achieved comparable performance with more sophisticated models. Different from TextCNN, Conneau et al. [6] proposed a CNN architecture that stacks lots of convolution layers deeply (e.g., twenty-nine layers) for sentence classification. They reported their very deep CNN model (VDCNN) shows better results than prior state-of-the-art methods. As a recent work, Le et al. [7] studied those two types of CNN models for sentence classification and sentiment analysis. They reported that a shallow-and-wide CNN model (e.g., TextCNN) shows comparable performance with VDCNN.

Based on their observation, we present a movie rating prediction architecture with a shallow-and-wide manner by extending TextCNN.

**Sentence embedding:** A basic form of the input for a neural network is a vector, and we need to convert the input data (e.g., picture and text) as a vector. There are various embedding schemes according to the type of inputs and the goal of the network. For sentence classification, morphological analysis is usually utilized to define a unit of embedding (i.e., a word) [8]. Generally, Part-Of- Speech (POS) tagging in Korean performs morphological analysis during POS tagging with considering the context of a sentence [9]. There are also several Korean POS taggers, such as Mecab-ko [10], Twitter [11], Hannanum [12], Etc.

In this work, we utilize Mecab-ko library since it is a well-organized [13], commonly used library.

**Attention mechanism:** Squeeze-and-Excitation (SE) block [14] was introduced to explicitly model inter-dependency between channels of its convolution features. This SE block re-calibrate channel-wise

feature with low computational overhead. As a result, they won the first place at ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2017 while significantly reduced the Top-5 error. From a wide point of view, the behavior of SE block is similar to the attention mechanism. SE block gives more attention to a channel that plays an important role while it gives less attention to a channel that does not. It learns the degree of attention by a small network.

Recently, Transformer was introduced in Vaswani et al. [15]. It is a self-attentional encoder-decoder network architecture and produced significant improvement in machine translation. BERT [16] is a universal language model based on Transformer and significantly outperformed existing methods in various NLP tasks. However, it requires huge computational and memory costs since Transformer is a very deep network architecture.

In this paper, we employ SE blocks to our network in two ways and analyze the benefits due to the lightweight computation cost with effectiveness.

## 3. Movie Rating Prediction Architecture

In this section, we introduce our movie rating prediction architecture that translates an audience's review written in Korean to a rating. We first describe the overall architecture (Sec. 3.1) and then explain our convolution layer design (Sec. 3.2). Finally, we introduce a novel scoring function for translating the result of the convolution layer to a rating (Sec. 3.3).

### 3.1 Overview

Our network consists of three components, including an embedding layer, convolution layers, and fully-connected layers (Fig. 2).

The embedding layer converts texts (i.e., audience's reviews) to vectors. We employ a character embedding scheme that handles the texts at a character level. This is because our target dataset contains un-normalized text (e.g., slags and incomplete sentences), and Yu et al. [17] showed that a character embedding scheme works better than a word embedding scheme for un-normalized texts. Once we get vectors, we take a spatial dropout stage to discard noise (e.g., unnecessary words or characters) in the raw texts. The spatial dropout operation selects elements of a vector randomly with a specific ratio and makes them to zero. This makes sense because our target dataset includes lots of meaningless single characters like 'ㅋㅋㅋ' or 'ㅎㅎㅎ' that means just laughing in Korean.

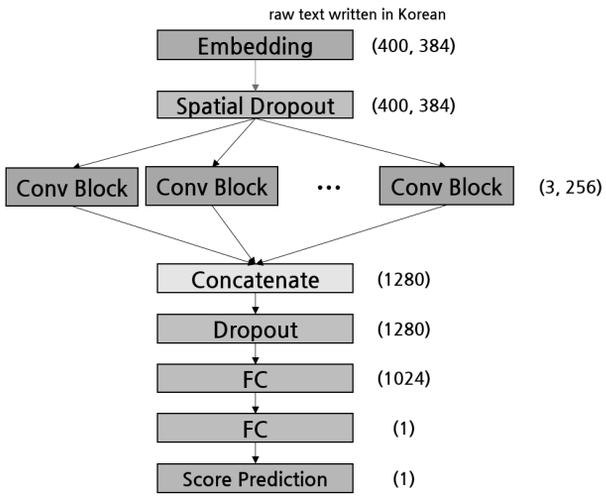


Fig. 2. Overview of Our CNN Architecture for Movie Rating Prediction

Character embeddings of a movie review are sent to the convolution layer. We design our convolution layer (Fig. 2) based on TextCNN [4]. The convolution layer consists of a set of convolution blocks that have different kernel sizes and work in parallel. This is a similar form of the CNN model used in TextCNN. However, we improve the representation power of a convolution layer by adding SE blocks (Sec. 3.2). Outputs of each convolution block (i.e., captured morphological information) are concatenated as a vector. Fully-connected layers compress the extracted features from the convolution layer. Finally, the compressed feature is translated as a rating with our scoring function (Sec. 3.3).

### 3.2 Convolution Layer

We design our basic convolution layer as a form of shallow-and-wide CNN model. To make a feature map, we put multiple convolution blocks having a different kernel size in a parallel manner and concatenate outputs of them. Each convolution block performs a 1D convolution operation and takes a rectifying stage. At the end of a convolution block, we use k-Max pooling to capture k ($k > 1$) features instead of max-over-time pooling used in

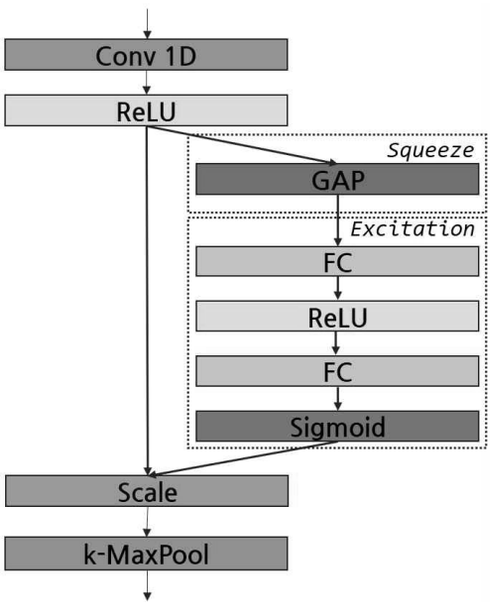


Fig. 3. Convolution Block in Our Network. We Improve the Basic Convolution Block with an SE Block (Dotted Box)

TextCNN. Also, we improve the basic convolution layer by adding SE blocks to focus on important features and increases the representational power of the network. A SE block consists of two operations, squeeze and excitation. The *squeeze operation* aggregates global spatial information using a global average pooling (GAP). This operation reduces the dimensionality of a feature to one by averaging outputs of each channel. The *excitation operation* adaptively re-calibrates feature maps of each channel by applying a gating mechanism for the output of the squeeze operation. The gating mechanism consists of two fully-connected (FC) layers that perform non-linear operations among channels. As discussed in SENets [14], the excitation operation is one of the major bottlenecks for training performance. To lessen such overhead, we compress the dimension of a channel from C to $\alpha$C (e.g., $\alpha$=1/16) in the first FC layer. Then, it passes the second FC layer and takes sigmoid activation. Finally, the result is scaled by channel-wise multiplication between the feature maps in the scale layer.

We apply this SE block in two ways. At first, we put a SE block in-between convolution layer and k-Max pooling layer in the basic convolution block, as shown in Fig. 3. With this approach, we aim to re-calibrate spatial correlation among features extracted with the same kernel size. The second approach is

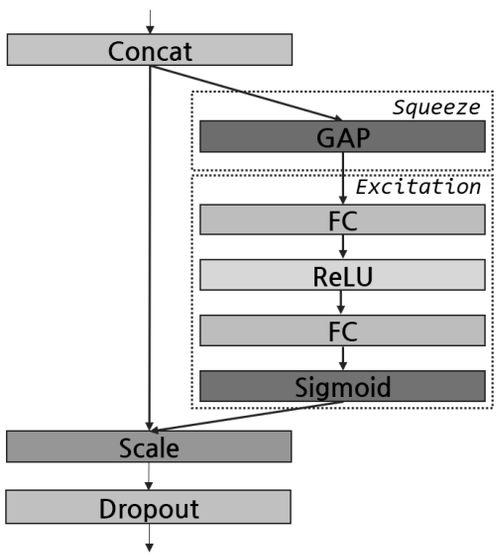


Fig. 4. Concatenate Layer in Our Network. The Dotted Boxes Show the SE Block We Add to the Basic Concatenate Layer

attaching a SE block after the concatenate layer (Fig. 4), and it catches the spatial information among features coming from convolution blocks having different kernel sizes.

### 3.3 Scoring Function

To predict a movie rating from the classification result of CNN models, we formulate a novel scoring function,

$$Score(x) = \frac{(\sigma(x) \times 9 + 11)}{2} \quad (1)$$

where $x$ is the output value from the last FC layer and means *tanh* function. It scales the results of the FC layer to the target rating ranged from 1 to 10. Instead of sigmoid function that usually employed for scaling value to a specific range, we use *tanh* function since it is more cost-effective than sigmoid function for training in the perspective from convergence speed. Table 1 compares the MSEs with our scoring functions with *sigmoid* and *tanh* on our prediction architectures.

Table 1. This Table Shows MSEs of Our Models with Scoring Functions Based on *Sigmoid* and *Tanh*

| Model | *sigmoid* | *tanh* |
|---|---|---|
| Ours-convSE | 3.8466 | 3.4043 |
| Ours-ConcatSE | 3.4786 | 3.3841 |

### 3.4 Training Loss

Prediction score ($\hat{y}$) and target score ($y$) are a real number ranging between 1 and 10. This is a simple regression problem with a closed range. Therefore, our loss function is defined as MSE between $\hat{y}$ and $y$:

$$Loss(\hat{y}, y) = \|\hat{y} - y\|_2 \quad (2)$$

## 4. Implementation

We have trained our network with Adam optimizer [18], and the mini-batch size is 128. To check the advantages of character embedding, we also have implemented our model with word embedding, and we have used the Adadelta optimizer [19] in this case since the usage rate of the variable varies according to the probability of occurrence of the word. We brought the character embedding method from a baseline code of NAVER A.I Hackathon 2018. Mecab-ko library was used to perform POS tagging reviews with morphological analysis [10]. To find the best embedding size, we did a greedy search and used 384 and 300 for character embedding and word embedding, respectively.

We used five convolution blocks in a parallel manner, and the convolution filter size is 10, 9, 7, 5, and 3 for character embedding. For the word embedding, we used four convolution blocks, and the convolution filter sizes are 2, 3, 4, and 5, respectively. The input dimension of each convolution filter is 256, and the dimension of outputs varies according to the filter size. A rectified linear unit (ReLU) with a threshold of 1e-6 was used as an activation function, k-MaxPooling was used with k of 3 for each convolution block. For two fully connected layers at the end of the network, we used 1024 and 1 unit(s), respectively.

The initial learning rate is 2e-4, and it is decreased by a factor of 0.95 for every 25,000 global steps. The dropping ratios for both the spatial drop after the embedding layer and the dropout layer after the concatenate layer are 0.7. We applied L2 regularization with a rate of 1e-3 for all the parameters on our network except the embedding layers. To prevent gradient overshooting, we clipped a norm of gradients by 5.

Table 2. This Table Shows the MSEs of Five Different Models Including Previous Work and Our Methods for the NAVER Movie Review Dataset

| Model | MSE |
|---|---|
| TextCNN-word | 11.3971 |
| TextCNN-char | 4.5875 |
| Ours-baseline | 4.1217 |
| Ours-convSE | 3.4043 |
| Ours-ConcatSE | **3.3841** |

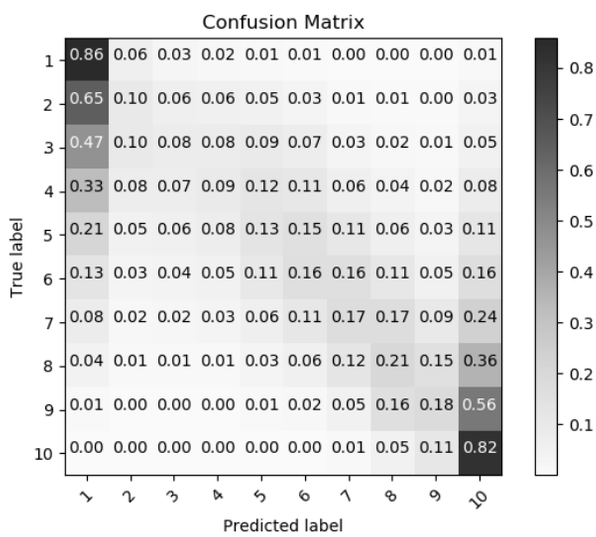


Fig. 5. This Confusion Matrix Shows the Prediction (or Classification) Performance of *Ours-ConcatSE*. To Draw This Matrix, We Rounded the Prediction Value

To compare the accuracy of our method, we have implemented three versions of our methods and two alternative models:

- **Ours-baseline** is our baseline model without any SE block (Fig. 2).
- **Ours-convSE** has implemented by adding the SE block to the convolution blocks (Fig. 3) to Ours-baseline.
- **Ours-concatSE** has implemented by attaching the SE block to the concatenate layer (Fig. 4) to Ours-baseline. Please note that we use the basic convolution block, not convSE, for this model.
- **TextCNN-word** is our implementation of the CNN model proposed by Kim [4]. In this model, we

Table 3. Distribution of Reviews in the NAVER Movie Dataset

| Rating (class) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # of reviews (millions) | 0.88 | 0.15 | 0.14 | 0.16 | 0.27 | 0.38 | 0.54 | 0.94 | 1.01 | 4.38 | 8.86 |
| Ratio (%) | 9.95 | 1.73 | 1.53 | 1.85 | 3.09 | 4.29 | 6.14 | 10.64 | 11.39 | 49.36 | 100.0 |

used word embedding and three convolution blocks whose kernel sizes are 3, 4, and 5, respectively. Different from our model, it does not include a spatial dropout layer while predicting the rating with our scoring function. For the rest of the hyper-parameters, we followed the paper.

- **TextCNN-char** has implemented by changing the embedding scheme to character embedding from the TextCNN-word.

We have implemented the models with TensorFlow 1.10, Numpy, and KoNLPy libraries.

## 5. Experiments

To check the accuracy of our movie rating prediction architecture, we have applied our method for NAVER Movie (Fig. 1). We made the NAVER movie review dataset by crawling reviews from the website. Each review data consists of a rating score ranged from 1 to 10, and reviews of an audience for a movie. The reviews are written in Korean, and many of them include un-normalized and incomplete sentences. The NAVER movie review dataset includes ten classes from 1 to 10 and contains 8.86 million reviews (Table 3). The average number of characters per review is 64, and the minimum and the maximum length is 1 and 299 character(s), respectively. We randomly split the dataset and have used 80% of them for training and others for validation.

Table 2 compares the MSEs of five different models. As shown in the results, character embedding greatly improves the prediction accuracy than word embedding for our dataset. This demonstrates that the character embedding scheme works better for un-normalized sentences than the word embedding scheme.

Our baseline implementation (i.e., Our baseline) shows better performance than TextCNN-char, and the accuracy is further improved with SE blocks. We found that both our approaches using SE blocks reduce the errors. This is because the SE blocks catch spatial correlation among extracted features well. In our experiment, Ours-concatSE achieves the lowest MSE, 3.3841, and it can be interpreted into the prediction results have errors higher or less than 1.83 in most case. These results demonstrate the accuracy of our method.

Fig. 5 is a confusion matrix that shows the prediction performance of Ours-ConcatSE. Each row shows the distribution of prediction results for a class (i.e., rating score). As shown in the confusion matrix, most of the prediction ratings are distributed around the true label. These results validate the usefulness of our CNN architecture for movie rating prediction.

## 6. Conclusion

We have presented a movie rating prediction architecture based on a CNN. It guesses a rating from an audience's reviews that are written in Korean and contains lots of slang and incomplete sentences. To handle such un-normalized texts efficiently, we utilized a character embedding scheme to cover many variants of words. Also, we employed SE blocks to catch spatial correlation among features. We also proposed a novel scoring function that converts the output of activation function to a score in a specific range (1-10). We have applied our architecture to a movie review dataset and have demonstrated the accuracy and usefulness of our method.

**Limitations and future work**: Although our method has achieved a low MSE, it shows relatively lower accuracy for medium-range (e.g., 4-7). To achieve more accurate prediction results over all ranges, we would like to improve our CNN model and scoring

function. In this paper, we have tested our architecture for only one dataset. As future work, we would like to apply our method to other datasets like other movie review datasets or reviews for other publications like books.

## References


[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, pp. 1097-1105, 2012.

[2] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22, 10, pp.1533-1545, 2014.

[3] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," In *Proceedings of the 25th Inter-National Conference on Machine Learning*, 160-167, 2008.

[4] Y. Kim, "Convolutional Neural Networks for Sentence Classification," ArXiv e-prints:1408.5882, 2014.

[5] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition," ArXiv e-prints:1409.1556, 2014.

[6] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very Deep Convolutional Networks for Text Classification," ArXiv eprints:1606.01781, 2016.

[7] H. T. Le, C. Cerisara, and A. Denis, "Do convolutional networks need to be deep for text classification?," arXiv preprint 1707.04108, 2017.

[8] C. M. Chang, J. Cho, H. Liu, R. K. Wagner, H. Shu, A. Zhou, C. S. Cheuk, and A. Muse, "Changing models across cultures: Associations of phonological awareness and morphological structure awareness with vocabulary and word recognition in second graders from beijing, hong kong, korea, and the united states," *Journal of Experimental Child Psychology*, Vol.92, No.2, pp.140-160, 2005.

[9] S. Petrov, D. Das, and R. McDonald. "A universal part-of-speech tagset," arXiv preprint arXiv:1104.2086, 2011.

[10] Mecab-ko morphological analyzer [internet], http://eunjeon.blogspot.com/.

[11] Twitter morphological analyzer [internet], https://openkoreantext.org.

[12] Hannanum morphological analyzer [internet], http://semanticweb.kaist.ac.kr/research/hannanum/index.html

[13] KoNLPy benchmark performance among the libraries, [internet] http://konlpy.org/en/latest/morph/#comparison-between-pos-tagging-classes.

[14] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-Excitation Networks," ArXiv e-prints:1709.01507, 2017.

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

[16] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2018.

[17] X. Yu, A. Faleńska, and N. T. Vu. A, "general-purpose tagger with convolutional neural networks," arXiv preprint:1706.01723, 2017.

[18] D. P. Kingma and J. Ba., "Adam: A Method for Stochastic Optimization," ArXiv e-prints, 1412.6980, 2014.

[19] M. D. Zeiler, "Adadelta: an adaptive learning rate method," arXiv preprint:1212.5701, 2012.



### Hyungchan Kim

https://orcid.org/0000-0002-1729-0580
e-mail : kozistr@gmail.com


He is an undergraduate student in the school of Computer Engineering at KOREATECH (Korea University of Technology and Education). His research interest is mainly designing the deep learning model, specially in the image domain, classification, semantic segmentation, super resolution.


### Heung-Seon Oh

https://orcid.org/0000-0002-9193-8998
e-mail : ohhs@koreatech.ac.kr


He is currently an assistant professor in the School of Computer Engineering at KOREATECH (Korea University of Technology and Education). He received his B.S. from Korea Aerospace University in 2006. He received his Ph.D. from KAIST (Korea Advanced Institute of Science and Technology) in Computer Science in 2014. His research interests are Deep Learning, Machine Learning, Natural Language Processing, and Computer Vision.

Duksu Kim

https://orcid.org/0000-0002-9075-3983

e-mail : bluekds@koreatech.ac.kr

He is currently an assistant professor in the School of Computer Engineering at KOREATECH (Korea University of Technology and Education). He received his B.S. from SungKyunKwan University in 2008. He received his Ph.D. from KAIST (Korea Advanced Institute of Science and Technology) in Computer Science in 2014. He spent several years as a senior researcher at KISTI National Super-computing Center. His research interest is designing heterogeneous parallel computing algorithms for various applications, including proximity computation, scientific visualization, and machine learning.