# Tensor Space Model based 2-Channel Text Classification with Contextual Embedding and the *Transformer* model*

SoonKwan Kwon, Han-Joon Kim+
*Department of Electrical and Computer Engineering*
*University of Seoul*
*Seoul, 02504 Republic of Korea*
*E-mail: ghghfhddl@naver.com; khj@uos.ac.kr*

Recently, the explosive increase in the amount of text data and the rapid spread of text generative models has caused automatic text classification for information (e.g., social data, review data, and fake news) to become increasingly important. This paper proposes a multimodal deep learning architecture that effectively combines contextual word embedding and the *Transformer* model under the tensor space representation model to achieve more reliable text classification. The tensor space representation model represents a single document as a *term*-by-*concept* matrix that contains the semantic information of words; however, it does not accommodate the polysemy problem or word sequence information. To achieve near-perfect text classification, we propose a two-channel deep learning architecture that can learn both word context information and word sequence information under a tensor space model. In our approach, the *Transformer* model is utilized to learn word sequence information; as a result, our proposed architecture produces a multimodal learning model for text classification. Using six textual datasets, we demonstrate the performance improvement of our proposed multimodal text classification architecture.

***Keywords:*** Deep learning, large language model, sentiment analysis, tensor space model, text classification, *Transformer*, word embedding

## 1. INTRODUCTION

The development of the Internet and the spread of smartphones have led to sharp increases in the use of social networking services, rapidly creating large amounts of textual data that are useful for understanding customers' information and needs. There has also been an increase in the availability of false information, such as fake news, due to the recent proliferation of artificial intelligence. In this era of textual big data, the ability to identify automatically reliable and unreliable information for text classification of information such as social data, review data, and fake news has become particularly important [1–7]. Neural network-based embedding techniques that generate embedding vectors with word semantic information have helped to greatly enhance text classification performance. However, early embedding techniques were not able to accommodate cases where one word had multiple meanings or slightly different meanings, depending on the context within a particular document. The latest embedding techniques, such as Embedding from Language Model (ELMo) [8] and Bidirectional Encoder Representations from Transformers (BERT) [9], have been better able to reflect the different meanings of individual words, depending on the context in the embedding vector.

+ Corresponding author: Han-joon Kim (Email: khj@uos.ac.kr, Tel: +822-6490-2339), University of Seoul, Korea
**Acknowledgments:** This work was supported by the 2022 Research Fund of the University of Seoul.

In our previous study, we improved text classification performance by converting the documents to be classified into a tensor space representation model that used Wikipedia pages for a concept space [10]. However, in the process of selecting words that help classifications to form within the term axis of the tensor space model, the sequence information of the words is lost. To overcome this problem, we propose a 2-channel deep learning architecture that can effectively learn semantic and context information, as well as sequence information, under the tensor space model. This approach uses contextual embeddings, such as ELMo or BERT vectors, to construct the concept axis of the tensor space model and adds *sequence channel*s constructed based on the *Transformer* model [11], allowing both sequence information and context information to be processed. To validate the performance of the proposed technique, we conducted various text classification experiments with six English textual datasets, including fake news and reviews; we present optimal hyperparameters that can be used to form the proposed deep learning architecture.

The rest of the paper is organized as follows. In Section 2, we introduce the background and related work of our proposed deep learning architecture for text classification. In Section 3, we describe the details of the proposed architecture. The results of the comparative experiments with existing machine learning models and the optimization process of our classification model are presented in Section 4. Finally, our conclusions and directions for future work are discussed in Section 5.

## 2. BACKGROUND AND RELATED WORK

### 2.1 Background

#### 2.1.1 Contextual Embeddings: ELMo and BERT

Embedding is a technology that represents unstructured data (e.g., words, sentences) into vectors that allow computers to understand the data. The embedding model is trained with a neural network, in which the embedding vectors of similar words are trained to be arranged closer to each other, whereas the embedding vectors of dissimilar words are trained to be arranged farther apart. Techniques such as Word2Vec [12] and Global Vectors for Word Representation (GloVe) [13] pretrain models with a large corpus to embed words. Context-based embedding techniques, such as ELMo and BERT, have also been developed with the ability to accommodate polysemy and context.

ELMo enables context-sensitive embedding by pretraining a bidirectional long short-term memory (LSTM) model that is capable of bidirectional information extraction on a large corpus. BERT works with contextual information by pretraining a model built using stacked *Transformer* blocks. Both models accept a sentence as input, and then output a word-level embedding vector that reflects the context.

#### 2.1.2 Tensor Space Model

The initial text representation model, bag of words (BoW) [14], utilizes the frequency of words in a document to represent it as a two-dimensional *document-by-term* matrix (DTM); that is, a single document represented as a vector. Subsequently, term frequency-

inverse document frequency (TF-IDF) [14] emerged as a model that considers the potential for the same word to have varying levels of importance in different documents. This method more accurately reflects word importance by assigning weights to the DTM according to the importance of words in the document. However, this method also has a significant drawback; although TF-IDF is a more effective representation model which solely utilizes frequency counts, it does not fully capture word meanings.

In contrast to the traditional model, we previously developed a semantic tensor space model that represents a document corpus as a three-dimensional tensor comprising concepts, terms, and documents (see Figure 1) [10]. This model enriches word meanings by adding a 'concept' axis to the *document-by-term* matrix. In other words, the tensor space model represents each document as not a *term* vector, but as a *concept-by-term* matrix that captures the relationship between term features and concept features. These matrices are then aggregated into a *concept-by-term-by-document* tensor. The initial tensor space model utilized informative Wikipedia pages to define individual semantic concepts for the concept axis. Subsequently, the advent of context-based word embedding techniques allowed the concept vector of the concept axis to be represented as a pretrained contextual embedding vector. As a result, this enhanced tensor space model, which integrates both semantic and contextual information, can improve text classification performance.

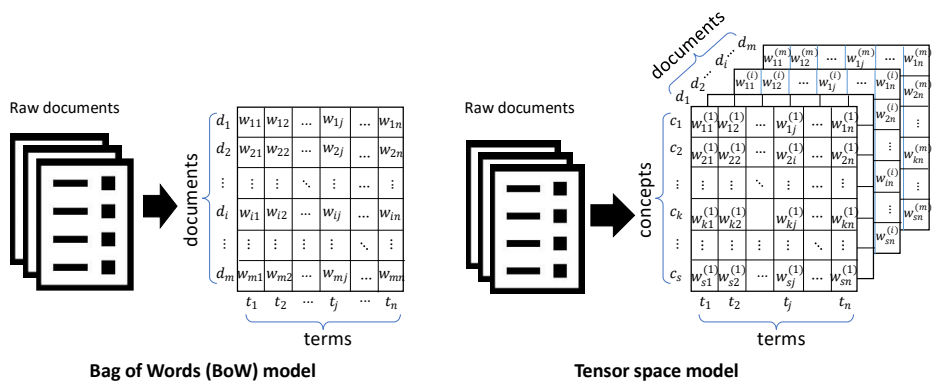


Fig. 1 Differences between BoW model and tensor space model: in the BoW model, $w_{ij}$ denotes the weight value of term $t_j$ document $d_i$. In the tensor space model, $w_{kj}^{(i)}$ denotes the weight value of each element corresponding concept $c_k$ of term $t_j$ in document $d_i$.

### 2.1.3 Attention Mechanism

Before explaining the attention mechanism, it is important to clarify the seq2seq [15] model. This model exhibits an encoder–decoder structure and is mainly used for machine language translation tasks. A limitation of this model is its basis on a recurrent neural network (RNN); when the length of the input sequence becomes long, gradient loss occurs and all information is compressed into a single context vector, resulting in information loss. The attention mechanism [16] is a method that emerged to resolve the above limitation of the seq2seq model. When predicting a decoder's output word, at each point in time, a query, key, and value structure are used to reference words in the encoder's input sequence that

are more related to the predicted word. The decoder's prediction performance is improved by focusing on important words that assist with prediction. The attention mechanism is used in natural language processing (NLP) tasks, such as translation, as well as various fields that utilize artificial neural networks, such as text classification and computer vision [17, 18].

#### 2.1.4 *Transformer*

Similar to the seq2seq model, the *Transformer* model consists of an encoder–decoder structure. The difference is that the encoder and decoder are not RNN-based; they are composed of sublayers such as position embedding, multi-head self-attention, and a position-wise feed-forward neural network. In the layered encoder and decoder stages, operations at each time step are performed in parallel, resulting in faster speeds and improved performance compared with RNN-based models. The *Transformer* model is used in machine language translation tasks, as well as text classification tasks and various NLP tasks [19–22].

### 2.2 Related work

As word embedding techniques have evolved, extensive research has been conducted regarding their uses in text classification. The TextCNN model proposed in [23] can effectively extract features of words using a convolutional neural network (CNN) applied to the word embedding matrix. In [1], the authors established an improved TextCNN method that adds a parts-of-speech (POS) tag channel containing POS information to the word embedding matrix and shows better performance on SemiEval-2014, 2015, and 2016 datasets. [5] proposed a more stable model for classification by constructing a multi-channel with various pretrained embedding vectors and utilizing a CNN and attention mechanisms to improve sentiment analysis performance. [2] argued that the importances of specific words vary among classes and proposed a multi-channel TextCNN model that applies a TF-IDF-based term weighting system to the TextCNN structure, thereby enhancing text classification performance.

There has also been research to improve text classification performance using Graph Convolutional Networks (GCN). In [24], TextGCN was proposed to improve text classification performance, and it represents the complex structure of text data in the form of a graph by representing documents and words as graph nodes and their relationships as edges. Following this, BertGCN was proposed as an enhanced model of TextGCN, integrating BERT with GCN [25]. It utilizes pre-trained BERT to represent documents as nodes and has achieved significantly high performance on several datasets through GCN training. As an alternative approach to utilizing pre-trained BERT, RoBERTa introduced a method for enhancing BERT by optimizing its pretraining process through the use of larger training datasets, extended training time, and fine-tuning of hyperparameters, which leads to improved text classification performance [26]. Subsequently, DeBERTa further improved RoBERTa by employing a disentangled attention mechanism and an enhanced mask decoder [27].

In addition, the emergence of Large Language Models (LLMs) [28] has recently stimulated research efforts aimed at enhancing text classification using a prompt-based

approach called In-Context Learning (ICL); ICL is a prompt engineering technique that boosts the performance of specific tasks by incorporating requirements and examples directly into the prompts. While ICL shares similarities with fine-tuning techniques in that it utilizes a pre-trained model, it differs in that it does not update the weights of the model during the learning process. The authors of [29] introduced 'Clue And Reasoning Prompting' (CARP) as an effective method for implementing ICL. They argued that a strong reasoning ability, grounded in complex language phenomena, is crucial for improving text classification performance. To achieve this, they demonstrated that extracting clues and reasoning from the document being classified can enhance classification results.

## 3. PROPOSED METHOD

### 3.1 Overview

As mentioned earlier, we proposed a three-dimensional tensor space model to achieve a more reliable text classifier in our previous study (see Section 2.1.2.). In the process of representing a document as *term-by-concept* matrix using the tensor space model, sequence information is lost. Thus, if the tensor space model was trained with a simple TextCNN structure, text classification was performed with semantic and context information. In this study, we have developed a 2-channel deep learning architecture that can effectively train semantic, context, and sequence information by adding a new channel that contains sequence information when classifying a contextual embedding-based tensor space model. Before explaining the proposed method in detail, the paragraph below explains the notation used in this paper.

A model that configures the concept axis of a tensor space model with a contextual embedding vector and classifies it into a simple TextCNN structure is referred to as a 1-channel tensor space model. The new 2-channel deep learning architecture that trains a channel consisting of contextual embedding vectors and a channel containing order information is referred to as a 2-channel tensor space model (see Figure 2). The 2-channel tensor space model includes a channel consisting of a context-based embedding matrix and a channel consisting of a *Transformer*-based positional embedding matrix that contains sequence information. The channel composed of the context-based embedding matrix utilizes the ELMo or BERT embedding vector, referred to as the ELMo channel or the BERT channel, respectively, depending on the embedding used. Another channel composed of a positional embedding matrix is the *sequence channel*. If the *context channel* and *sequence channel* are used in a 2-channel tensor space model, then the model is referred to as '2-Channel Tensor (ELMo + Sequence)'; if the BERT channel is used instead of the ELMo channel, it is denoted as '2-Channel Tensor (BERT + Sequence)'. Similarly, the 1-channel tensor space model is written as '1-Channel Tensor (ELMo)' or '1-Channel Tensor (BERT)', depending on the embedding used.

In Sections 3.2–3.4, we describe in detail how the 2-channel tensor space model is constructed and trained.

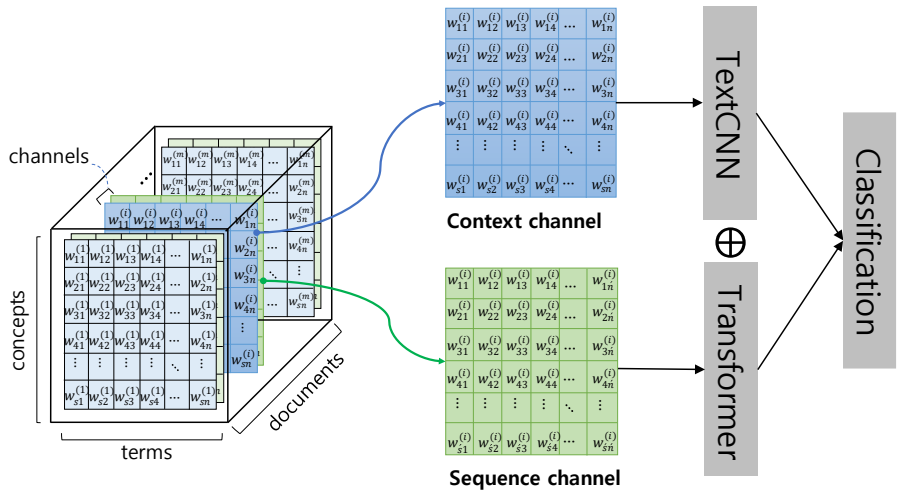


Fig. 2. Overview of 2-channel tensor space model

### 3.2 Conversion to the context channel

To construct the *context channel*, it is necessary to first pretrain context-based embeddings ELMo (or BERT). In the case of ELMo, pretraining is performed using the dataset to be classified, and the embedding vector output is set to 256 dimensions to reduce computational costs. For BERT, the pretrained model within the Transformer package provided by Hugging Face is used and its size of embedding vector is set to 768.

The documents in the dataset are embedded using the trained embedding model. When embedding with ELMo, embedding vectors for each word appearing in the document can be obtained. For example, suppose the input document is ‘This is a multimodal deep learning architecture’. If this document is input into pretrained ELMo embedding, it would be represented by seven 256-dimensional vectors corresponding to ‘This’, ‘is’, ‘a’, ‘multimodal’, ‘deep’, ‘learning’ and ‘architecture’. Embedding the documents of the dataset to be classified allows for the creation of a word embedding matrix that consists of the embedding vectors for words within the document.

In the case of embedding documents with BERT, the difference is that BERT outputs embedding vectors for subtokens, rather than for each word included in the document. Because the concept axis of the tensor space model requires word-level embedding vectors, the average value of the subtoken embedding vectors for a word is defined as the word embedding vector. For example, if the word ‘embeddings’ exists within a document, it would be split into four subtokens: ‘em’, ‘##bed’, ‘##ding’, and ‘##s’, producing four 768-dimensional embedding vectors. A single 768-dimensional vector, obtained by averaging these four vectors, becomes the embedding vector for the word ‘embeddings’. In this manner, a word embedding matrix for the words within the document can also be obtained with BERT embeddings. The following explanation proceeds with trained ELMo as an example, but the same process is applied in the case of BERT.

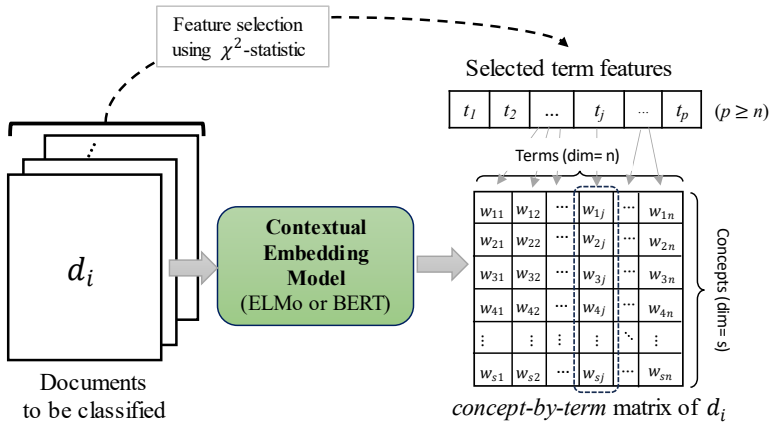


Fig. 3. The process of converting a document into a *context channel*

The next step involves identifying the significant words that will constitute the term axis of the *context channel*. For each dataset to be classified, stop words are removed, and *k* words that aid in text classification based on $\chi^2$ statistic are selected. Among the documents in the dataset to be classified, the document containing the highest number of selected words is identified, and the number of selected words in that document is defined as the maximum number of features (denoted as *m*). For each document, a matrix is created to become the initial *context channel*, where the column length equals the maximum number of term features, the row length equals the size of the embedding vector (e.g., 256), and all values are set to zero (see Figure 3). If a document contains any of the selected words, the contextual embedding vector for the selected word from the previously obtained word embedding matrix is found and sequentially mapped from the first row in the initial *context channel* matrix. This constructed *context channel* contains contextual and semantic information about the words selected based on the $\chi^2$ statistic.

### 3.3 Conversion to the sequence channel

The structure of the *sequence channel* is identical to the position embedding matrix of the *Transformer* encoder. The sequence of words that appears in each document undergoes integer encoding twice, and the maximum length of the sequence of words is set by the user to determine the input size. In the first integer encoding, indices are assigned based on the frequency of occurrence in the word set, and zero padding is applied according to the predetermined maximum length. The second integer encoding assigns indices in order from 0 to the maximum length, based on word position. The integer-encoded words are then transformed into two embedding vectors per word through an embedding layer. These embedding vectors are referred to as the *token embedding vector*, which reflects the meaning of the word, and the *position embedding vector*, which contains positional information. Both the *token embedding vector* and *position embedding vector* are updated during the training process, allowing the model to learn the semantic information of words and their sequence information. The sum of these two embedding vectors becomes one embedding

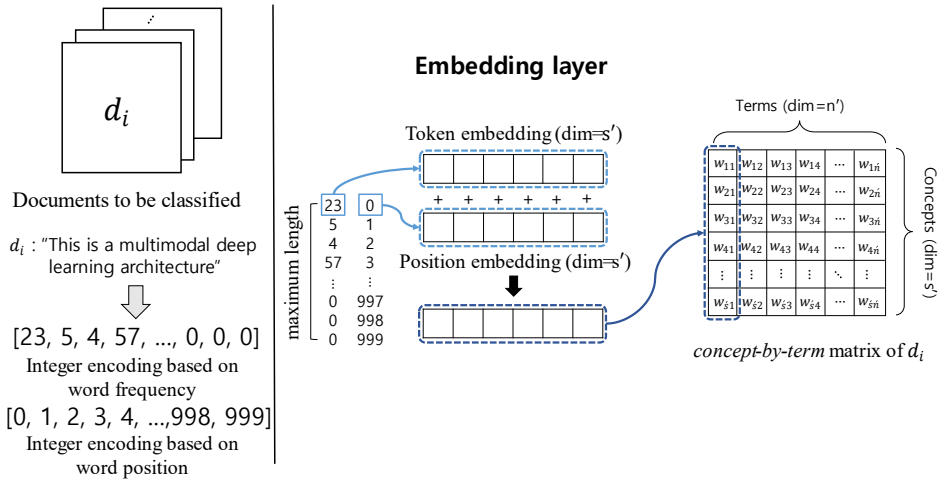


Fig. 4. The process of converting a document into a *sequence channel*

vector that corresponds to a single word. The embedding vectors representing each word within the sequence of words are gathered to form an embedding matrix, which constitutes the *sequence channel* (see Figure 4). This constructed *sequence channel* contains both the sequential and semantic information of the sequence of words.

### 3.4 Tensor space model-based 2-channel deep learning architecture

When the documents to be classified are transformed into the *context* and *sequence channels*, the 2-channel tensor space model is trained to effectively process the information contained in each channel. The *context channel* is trained with a TextCNN structure, whereas the *sequence channel* is trained with a *Transformer* structure. The feature maps trained from both structures pass through a self-attention layer, where important elements are assigned weighted to improve text classification performance. Feature maps that pass and do not pass through the self-attention layer are concatenated into one feature map and finally classified into a fully-connected layer. However, in certain datasets (Word Embedding over Linguistic Features for Fake News Detection (WELFake), 20Newsgroups, and R8), the application of self-attention to the trained features does not improve classification performance; thus, its application is selectively determined based on the dataset.

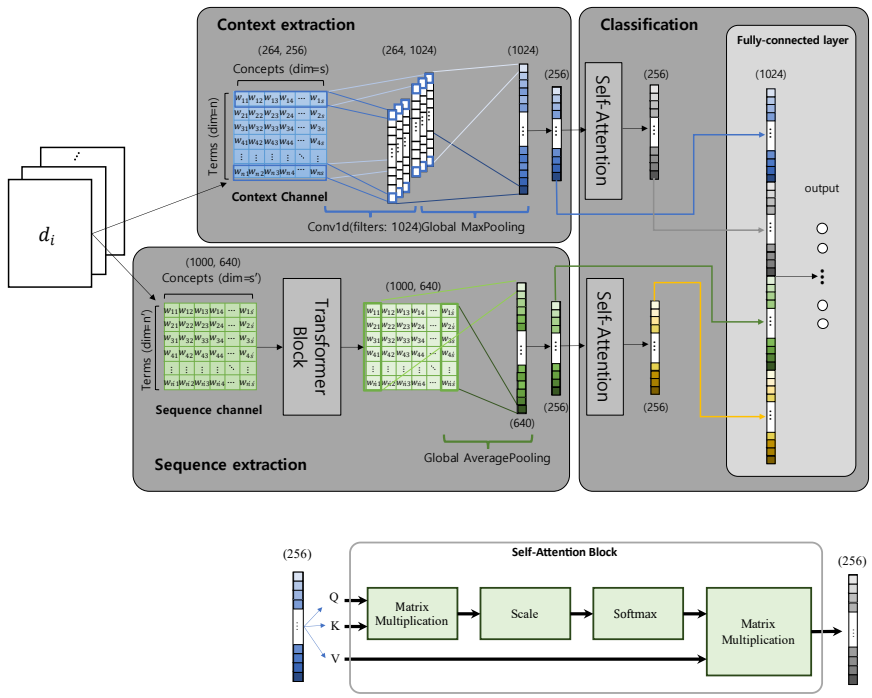


Fig. 5. Overall text classification architecture with a 2-channel tensor space model: it consists of three parts: context extraction, sequence extraction, and classification. The *context channel* and *sequence channel* are each trained with a TextCNN structure and a Transformer structure to form abstracted feature maps, then combined through a self-attention layer. The combined feature maps are finally fed into a fully-connected layer. For convenience, we use $w_{jk}$ instead of $w_{jk}^{(i)}$ to denote the weight value of the element corresponding to concept $c_k$ of term $t_j$ in document $d_i$.

Figures 5, 6, and 7 show the 2-channel tensor space model in detail, using the IMDB dataset (one of the datasets utilized to measure model performance in Section 4) as an example. As shown in the figures, the proposed text classification architecture with the 2-channel tensor space model is composed of three parts: 'context extraction', 'sequence extraction', and 'classification'. In the Self-Attention block of Figure 5, matrix multiplication is first performed between inputs Q and K, and in the scaling step, normalization is carried out by considering the length of the concept dimension (or the embedding size). After passing through the Softmax function, the result is multiplied by input V to produce the final output. As seen in Figure 5, the intermediate feature map (vector) is simultaneously fed into Q, K, and V, ultimately transforming it into a new feature map (vector) that weighs the values of elements aiding in text classification.

First, the context extraction part shows the details of the process where the *context channel* is trained using a TextCNN structure. A convolution operation is performed once with the *context channel* as input. The shape of each filter used in the convolution operation is (1, 256), and the number of filters is 1024. The 1024 feature maps produced by the convolution layer are represented as a 1024-dimensional vector through global max-

pooling. This 1024-dimensional vector is passed through a fully-connected layer, where the contextual and semantic information are included in a 256-dimensional abstracted feature.

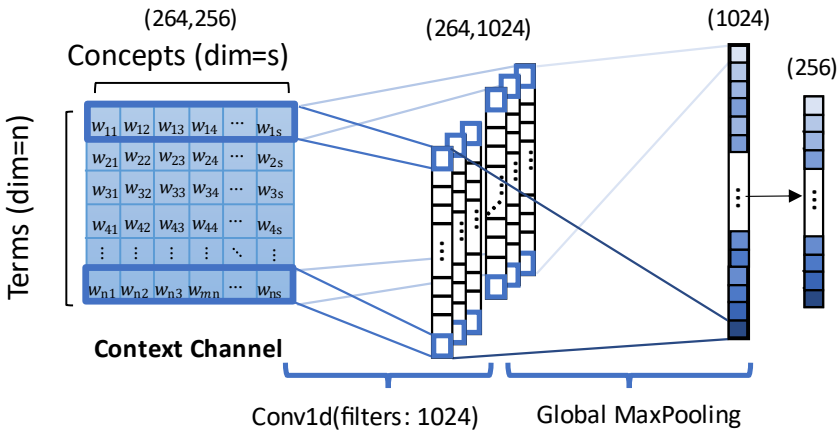


Fig. 6. The context extraction part: the *context channel* is trained with the TextCNN structure, and feature map vectors containing semantic information and context information are extracted.

The sequence extraction part shows the details of the process where the *sequence channel* is trained using a *Transformer* encoder structure. The *sequence channel*, as an input, passes through the *Transformer* block (see Figure 7). When the embedding vector of each time step is input to the *Transformer* block, it passes through a multi-head attention layer. Because the number of heads is set to 1, the process is identical to passing through the self-attention layer. Similar to the original *Transformer* model, a residual connection that sums input and output values is utilized to reduce information loss during training, and layer normalization is conducted to improve generalization performance. The embedding vector, which has passed through the self-attention layer, undergoes dimensionality reduction, and is then restored to its original dimensions. During this process, it passes through the fully-connected layer twice. Residual connection and layer normalization are also performed in this process. The output of the *Transformer* block is flattened through a global average-pooling layer to form the same number of nodes as the number of embedding dimensions (denoted as $s'$). These nodes pass through a fully-connected layer again and become 256-dimensional abstracted features that include sequence information and semantic information.

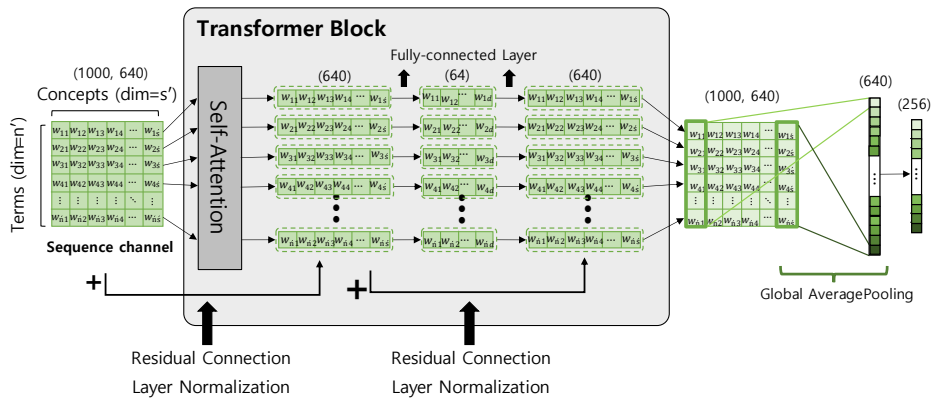


Fig. 7. The sequence extraction part: the *sequence channel* is trained with the *Transformer* encoder structure, and feature map vectors containing semantic information and sequence information are extracted.

Both feature maps trained from different models become weighted feature maps as they pass through the self-attention layer. Feature maps that have and have not passed through the self-attention layer are all concatenated to form a single 1024-dimensional vector. This is classified as the class with the highest probability through a fully-connected layer and softmax function.

## 4. EXPERIMENTS

### 4.1 Experimental datasets

Table 1. Datasets: to proceed smoothly with the experiment, duplicates and missing values were removed from the original samples of the dataset and sampled to suit the memory environment.

| Dataset | #Sample (original) | #Class (original) | Training data | Test data | #Class (experiment) |
|---|---|---|---|---|---|
| IMDB | 50,000 | 2 | 16,000 | 4,000 | 2 |
| WELFake | 72,134 | 2 | 16,000 | 4,000 | 2 |
| 20Newsgroups | 18,828 | 20 | 15,062 | 3,766 | 20 |
| R8 | 7,673 | 8 | 5,484 | 2,189 | 8 |
| News Category | 209,527 | 42 | 8,000 | 2,000 | 10 |
| AG News | 127,600 | 4 | 16,000 | 4,000 | 4 |

To assess the performance of the proposed model, we utilized the sentiment analysis benchmark Internet Movie Database (IMDB) dataset and the WELFake dataset for evaluating fake news detection efficiency. Additionally, we measured news topic classification performance using four English news datasets (20Newsgroups, R8, News Category, and AG News). All datasets included in the experiments were subjected to text preprocessing

steps, including special character handling and conversion to lowercase. Each dataset was configured to ensure smooth experimentation within the particular memory constraints. For datasets with an excessively large number of samples, we adjusted the sizes of the training and test data. If a dataset contained a class with insufficient samples, that class was excluded (or its balance was adjusted) before experiments were performed. As a result, the training and test data were divided at approximately an 80:20 ratio. The details of datasets used in the experiments are provided in Table 1.

### 4.2 Experimental setup

We classified the datasets introduced in Table 1 using the proposed model and existing models, then compared their performances according to accuracy. The models for comparison included: 1-Channel Tensor (ELMo), 1-Channel Tensor (BERT), 2-Channel Tensor (ELMo), 2-Channel Tensor (BERT), and conventional machine learning algorithms (such as support vector machine, Naïve Bayes, and logistic regression). The various hyperparameters for each dataset and model are described in Figures 8, 9, 10 and Table 2. For convenience, the function names are written using Python terminology.

For this experiment, we trained the proposed models and other models using RTX 4070 Ti GPUs, an Intel i9 (32 core) CPU, and 64G of RAM. We observed that the training time varied significantly across different datasets. For simpler datasets, our model typically completed training in under 5 minutes, while more complex datasets required between 15 and 20 minutes for 50 epochs. Regarding inference speed, our model consistently achieved prediction times of under 30 seconds.

#### 4.2.1 Hyperparameter Setup for 1-Channel Tensor Space Model

The 1-channel tensor is a model that classifies a tensor space model consisting only of contextual embedding channels into the TextCNN structure. The architecture of the model takes the form of adding a fully-connected layer to Figure 6; this approach classifies the output into the corresponding prediction class. In the convolution operation that accepts the contextual embedding channel of the tensor space model as input, the size of the filter was (1, embedding size), and the number of filters was set differently for each dataset. The ratio of dropout in the Dropout-①,② layers was set to 0 or 0.5, the activation function of the output layer was softmax, each of the other activation functions was ReLU, and the batch size was set to 256. For detailed dataset-specific hyperparameters, refer to Figure 8 and Table 2.

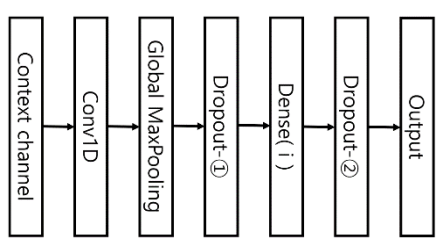


Fig. 8. 1-Channel Tensor conceptual diagram for hyperparameter setting.

### 4.2.2 *Transformer* Classifier Hyperparameter Setup

The *Transformer* classifier is a classification model that utilizes the encoder structure of the *Transformer* model. As in Section 4.2.1, the architecture of the model adds a fully-connected layer to the structure shown in Figure 7, which classifies the output into the corresponding predicted class. The Dropout-①,② ratios within the *Transformer* block were set to 0.1, identical to the existing *Transformer* model; the other Dropout-③,④ ratios were set to 0 or 0.5. The dimensionality of the embedding vector of the *sequence channel* was set differently for each dataset; Dense(i) and Dense(iii) were also set differently for each dataset. Dense(ii) is the point where the reduced embedding vector dimension is restored; thus, it was set to the same dimensionality as the input embedding. The activation function of the output layer was set to softmax, each of the other activation functions was set to ReLU, and the batch size was set to 256. For detailed dataset-specific hyperparameters, refer to Figure 9 and Table 2.

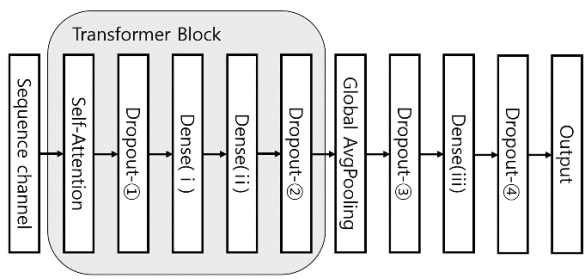


Fig. 9. *Transformer* classifier conceptual diagram for hyperparameter setting.

### 4.2.3 2-Channel Tensor Hyperparameter Setup

The settings of the convolution operation applied in the proposed model, the 2-channel tensor, were set identically to those of the aforementioned 1-channel tensor. Additionally, the hyperparameter settings that varied for each dataset were the dimensionality of the embedding vector in the *sequence channel*, the number of dense function nodes excluding Dense(iii), and whether self-attention was applied to the trained feature. In the case of dropout, the ratio within the *Transformer* block was set to 0.1 and the rest were set to 0 or 0.5, as in the *Transformer* classifier described above. Similarly, the number of nodes in Dense(iii) was equal to the dimensionality of the embedding vectors of the *sequence channel*. The activation function of the output layer was set to softmax, each of the other activation functions were set to ReLU, and the batch size was set to 256. For detailed dataset-specific hyperparameters, refer to Figure 10 and Table 2.

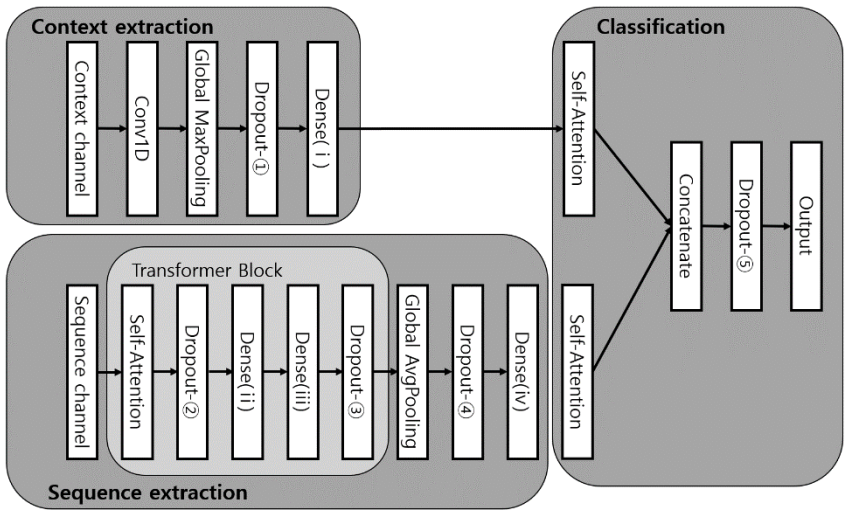


Fig. 10. 2-channel tensor conceptual diagram for hyperparameter setting.

### 4.2.4 Hyperparameter Optimization

To determine the optimal hyperparameters for the deep learning model, we prioritized the hyperparameters to be varied, establishing the best settings for a particular hyperparameter, and then moving on to explore the next hyperparameter for optimization. This approach was adopted due to the time and cost burden of considering all possible combinations of hyperparameters. For our proposed 2-channel tensor space model, we explored and established the optimal values for applying Dropout-①,④,⑤ to either 0 or 0.5 through grid search. Next, we examined performance variations according to the presence or absence of a self-attention layer before concatenation. The remaining continuous hyperparameters were changed to powers of 2 and explored in the order of the dimensionality of the position embedding vector, the number of Conv1D filters, and the number of dense layer nodes. A secondary optimization process was conducted to determine the precise optimal value for dimensionality of the embedding vector in the *sequence channel*. With the other hyperparameters at specific values, the dimensionality of the embedding vector was increased from 128 in intervals of 128 up to 1280 to measure performance. For the IMDB, WELFake, and 20Newsgroups datasets, due to memory limitations, the dimensionality of the embedding vector was increased from 64 to 640 (in intervals of 64) to measure performance. The optimal hyperparameters of the ultimately identified deep learning models are presented in Table 2.

Table 2. Datasets: Hyperparameter settings: the meaning of each model is described in the table footer; the locations of options for column-specific models are presented in Figures 7, 8, and 9. ('Self-Attention' refers to whether self-attention is performed in the classification step of the 2-channel tensor. 'Position Embedding' refers to the dimensionality of the position embedding vector. 'Conv1D # of filter' refers to the number of filters in the convolution operation performed in a 1-channel or 2-channel tensor.)

| Dataset | Model | Dropout ① | Dropout ② | Dropout ③ | Dropout ④ | Dropout ⑤ | Self-Attention | Position Embedding | Conv1D # of filter | Dense (i) | Dense (ii) | Dense (iii) | Dense (iv) | Epoch |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMDB | A | 0.5 | 0 | - | - | - | - | - | 1024 | 256 | - | - | - | 15 |
| | B | 0.5 | 0.5 | - | - | - | - | - | 1024 | 256 | - | - | - | 15 |
| | C | 0 | 0.1 | 0.1 | 0.5 | 0.5 | T | 640 | 1024 | 256 | 64 | 640 | 256 | 20 |
| | D | 0 | 0.1 | 0.1 | 0.5 | 0.5 | T | 640 | 1024 | 256 | 32 | 640 | 256 | 45 |
| | E | 0.1 | 0.1 | 0.5 | F | - | - | 64 | - | 32 | 64 | 256 | - | 5 |
| WELFake | A | 0.5 | 0.5 | - | - | - | - | - | 1024 | 256 | - | - | - | 20 |
| | B | 0.5 | 0.5 | - | - | - | - | - | 1024 | 256 | - | - | - | 20 |
| | C | 0 | 0.1 | 0.1 | 0.5 | 0 | F | 192 | 1024 | 256 | 128 | 192 | 256 | 10 |
| | D | 0 | 0.1 | 0.1 | 0.5 | 0 | F | 512 | 512 | 256 | 32 | 512 | 256 | 5 |
| | E | 0.1 | 0.1 | 0.5 | F | - | - | 128 | - | 128 | 128 | 256 | - | 5 |
| 20News-groups | A | 0.5 | 0 | - | - | - | - | - | 256 | 256 | - | - | - | 50 |
| | B | 0.5 | 0 | - | - | - | - | - | 512 | 1024 | - | - | - | 45 |
| | C | 0 | 0.1 | 0.1 | 0.5 | 0.5 | F | 384 | 256 | 256 | 32 | 384 | 256 | 10 |
| | D | 0 | 0.1 | 0.1 | 0.5 | 0.5 | F | 512 | 1024 | 1024 | 32 | 512 | 256 | 10 |
| | E | 0.1 | 0.1 | 0.5 | 0.5 | - | - | 256 | - | 32 | 256 | 256 | - | 50 |
| R8 | A | 0.5 | 0.5 | - | - | - | - | - | 512 | 256 | - | - | - | 35 |
| | B | 0.5 | 0.5 | - | - | - | - | - | 1024 | 1024 | - | - | - | 45 |
| | C | 0.5 | 0.1 | 0.1 | 0.5 | 0 | F | 256 | 256 | 256 | 32 | 256 | 256 | 15 |
| | D | 0 | 0.1 | 0.1 | 0.5 | 0.5 | F | 256 | 256 | 256 | 32 | 256 | 56 | 20 |
| | E | 0.1 | 0.1 | 0.5 | 0 | - | - | 256 | - | 64 | 256 | 256 | - | 25 |
| News Category | A | 0.5 | 0 | - | - | - | - | - | 1024 | 1024 | - | - | - | 20 |
| | B | 0.5 | 0 | - | - | - | - | - | 1024 | 1024 | - | - | - | 20 |
| | C | 0 | 0.1 | 0.1 | 0.5 | 0.5 | T | 640 | 1024 | 256 | 32 | 640 | 256 | 45 |
| | D | 0 | 0.1 | 0.1 | F | 0.5 | T | 768 | 1024 | 256 | 64 | 768 | 256 | 45 |
| | E | 0.1 | 0.1 | 0.5 | 0.5 | - | - | 512 | - | 64 | 512 | 512 | - | 20 |
| AG News | A | 0.5 | 0 | - | - | - | - | - | 256 | 256 | - | - | - | 50 |
| | B | 0.5 | 0 | - | - | - | - | - | 512 | 1024 | - | - | - | 45 |
| | C | 0 | 0.1 | 0.1 | 0.5 | 0.5 | T | 640 | 1024 | 256 | 32 | 640 | 256 | 30 |
| | D | 0 | 0.1 | 0.1 | 0.5 | 0.5 | T | 384 | 1024 | 1024 | 32 | 384 | 256 | 10 |
| | E | 0.1 | 0.1 | 0.5 | 0.5 | - | - | 256 | - | 32 | 256 | 256 | - | 50 |

A: 1-Channel Tensor (ELMo)

B: 1-Channel Tensor (BERT)

C: 2-Channel Tensor (ELMo)

D: 2-Channel Tensor (BERT)

E: *Transformer* classifier

The hyperparameter optimization for conventional machine learning algorithms such as SVM, Naïve Bayes, and logistic regression was conducted through grid search. For SVM and logistic regression, we varied the C value from 0.1 to 1000 in powers of 10, then explored both L1 and L2 penalties. For Naïve Bayes, we varied the alpha value from 0.01 to 100 in powers of 10.

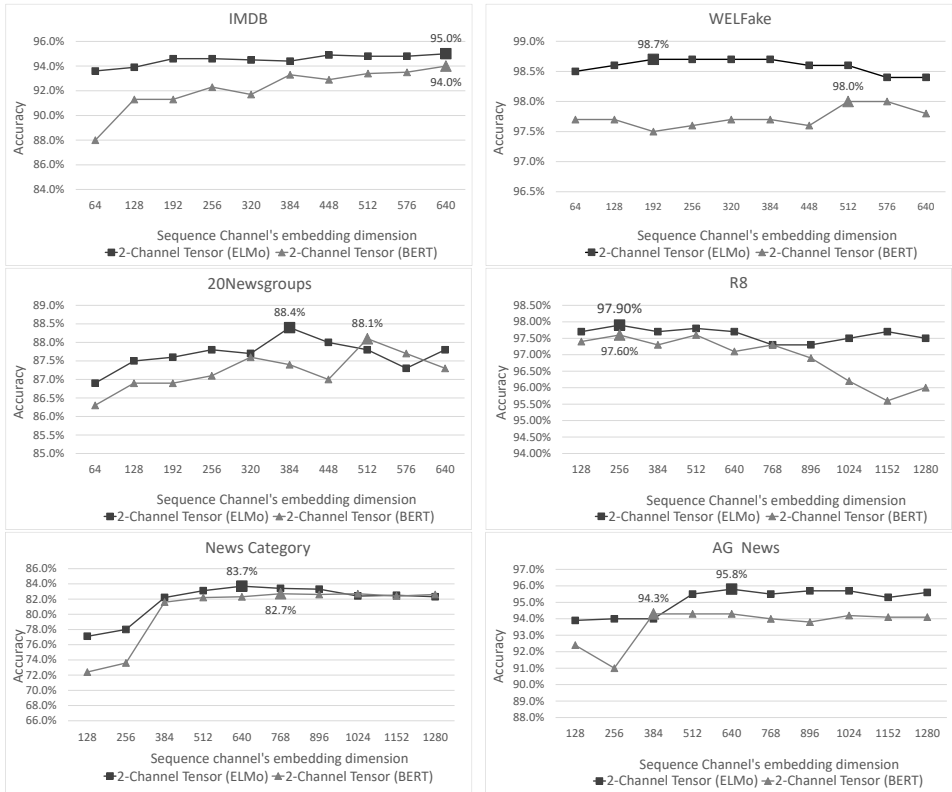


Fig. 11. The classification accuracy according to dimensionality: To determine the optimal dimensionality of the embedding vectors in the *sequence channel*, the classification accuracy according to dimensionality was visualized using a line graph. The point corresponding to the optimal value indicates the classification accuracy and is represented as a shape larger than the surrounding vertices. IMDB: Internet Movie Database; WELFake: Word Embedding over Linguistic Features for Fake News Detection.

### 4.3 Experimental results

Table 3 presents the performance evaluation of the proposed model and other models using six English text datasets. The best performing model in each dataset is highlighted in bold. When the baseline model for the proposed 2-channel tensor was set to a 1-channel tensor, improvements in performance were observed across all six datasets. In the case of WELFake and R8 datasets, the performance improvement of the 2-channel tensor was minimal compared with the 1-channel tensor; however, in the other datasets, the performance improvement of the proposed model was noticeable. Specifically, in the IMDB, News Category, and AG news datasets, the performance of the 2-channel tensor showed a substantial difference compared with the second-ranked model. The performance gaps between the 2-channel tensor and the second-ranked model were 4% in the IMDB dataset, 3.8% in the AG news dataset, and 8.8% in the News Category.

In the experiments, classification models using ELMo outperformed those using BERT. This is likely because the tensor space model necessitates word-level embedding

vectors. That is, since ELMo directly provides word-level embeddings, it seems better suited for tensor space models compared to BERT, which derives word-level embeddings from subword token-level vectors.

Table 3. Performance comparison between the proposed models and other machine learning models in terms of classification accuracy: the performance of the best model for each dataset is indicated in bold.

| Model | Dataset | | | | | |
|---|---|---|---|---|---|---|
| | IMDB | R8 | WELFake | News Category | 20Newsgroups | AG news |
| 1-Channel Tensor (ELMo) | 91.0% | 97.7% | 98.4% | 74.9% | 84.3% | 91.8% |
| 1-Channel Tensor (BERT) | 90.5% | 96.8% | 94.2% | 75.9% | 80.3% | 91.6% |
| 2-Channel Tensor (ELMo) | **95.0%** | **97.9%** | **98.7%** | **85.1%** | **88.4%** | **96.1%** |
| 2-Channel Tensor (BERT) | 94.0% | 97.6% | 98.0% | 84.5% | 88.1% | 94.5% |
| Transformer classifier | 88.9% | 97.2% | 96.4% | 65.5% | 87.4% | 89.1% |
| SVM | 87.2% | 97.3% | 94.5% | 71.0% | 86.6% | 91.3% |
| Naïve Bayes | 86.4% | 95.3% | 87.4% | 71.7% | 86.3% | 92.0% |
| Logistic Regression | 87.2% | 96.7% | 94.2% | 69.2% | 86.3% | 91.3% |

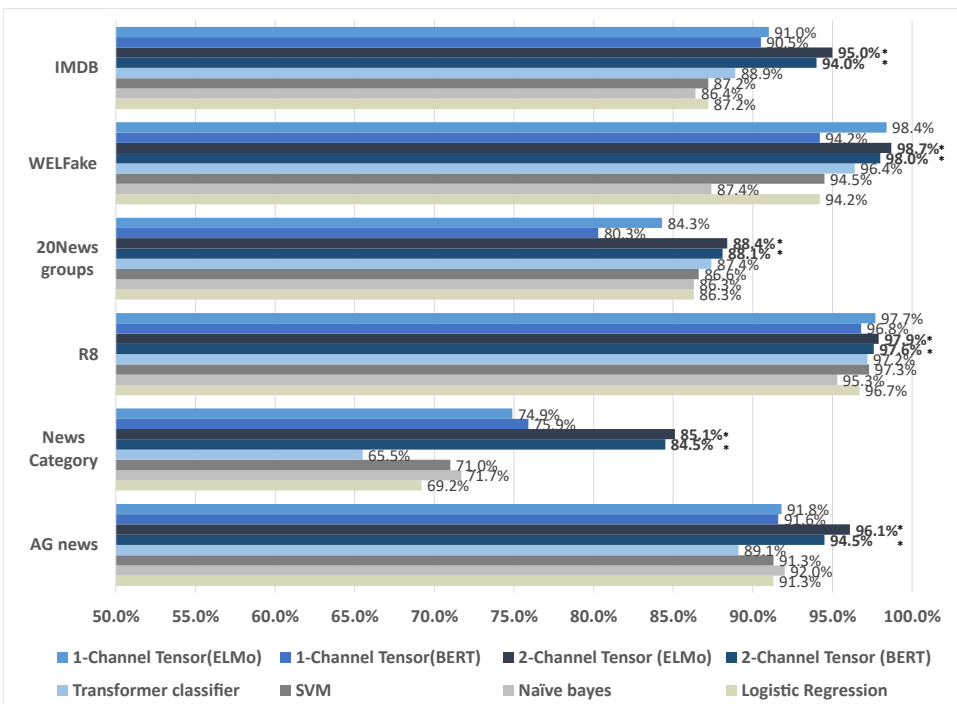


Fig. 12. Bar chart illustrating the performance comparison in Table 3: For the proposed model, an asterisk (*) is placed on the right side of the bar.

### 4.4 Performance comparison with LLM-based text classification methods

In order to compare with recent state-of-the-art techniques that have demonstrated excellent classification performance using large language models (LLMs), this section presents a performance comparison with the existing LLM fine-tuning methods (such as RoBERTa-Large [26], DeBERTa [27], and BertGCN [25]) and in-context learning (ICL) methods (such as GPT-3 Vanilla [29], GPT-3 CARP [29], and GPT-3 CoT [30]). The datasets used here are AG News, R8, which are common with this study and [27]. Table 5 shows the classification accuracy and parameter size of each model for the two datasets.

The parameter size is calculated as the sum of the parameters from the pre-trained language model and those from the classification model. The fine-tuning method using RoBERTa-Large, proposed in [28], has a total of 355 million parameters. DeBERTa, introduced in [29], features 134 million parameters in its DeBERTa-based model. The size of additional parameters trained during the fine-tuning process for each language model in the text classification task are denoted as α. Similarly, the parameter size for the proposed method is derived from the sum of 14 million parameters from ELMo (embedding size of 256) and 110 million parameters from BERT, along with the parameters trained for the classification model.

As shown in Table 5, for the AG News dataset, GPT-3 CARP slightly outperforms the proposed model, while for the R8 dataset, both BertGCN and GPT-3 CARP demonstrate better performance. When considering both parameter size and classification accuracy, the proposed model using a 2-channel tensor with ELMo achieves results that are close to state-of-the-art, despite having a significantly smaller number of parameters. Note that the 2-channel tensor with ELMo has a parameter size of just 46M and 19M for the AG News and R8 datasets, respectively.

Table 5. Performance comparison between the proposed models and recent LLM-based models in terms of classification accuracy and parameter size (M: Million, B: Billion): the performance of the proposed models and those of models that outperform the proposed models are indicated in bold.

| Dataset | Model | | Accuracy (%) | Parameter Size |
|---|---|---|---|---|
| AG News | Fine-tuning method | RoBERTa-Large | 95.6% | 355M+α |
| | | DeBERTa | 95.3% | 134M+ α |
| | | BertGCN | 95.7% | 111M |
| | In-context learning method | GPT-3 Vanilla | 94.1% | 175B |
| | | GPT-3 CoT | 94.9% | 175B |
| | | GPT-3 CARP | **96.4%** | 175B |
| | Proposed method | 2-Channel Tensor (ELMo) | **96.1%** | 46M |
| | | 2-Channel Tensor (BERT) | **94.5%** | 130M |
| R8 | Fine-tuning method | RoBERTa-Large | 97.8% | 355M+α |
| | | DeBERTa | 98.3% | 134M+α |
| | | BertGCN | **98.1%** | 111M |
| | In-context learning method | GPT-3 Vanilla | 95.6% | 175B |
| | | GPT-3 CoT | 95.6% | 175B |
| | | GPT-3 CARP | **98.9%** | 175B |
| | Proposed method | 2-Channel Tensor (ELMo) | **97.9%** | 19M |
| | | 2-Channel Tensor (BERT) | **97.6%** | 115M |

## 5. CONCLUSIONS

To improve text classification performance, the classification architecture should be trained so as to account for semantic, contextual, and sequential information. In our study, we incorporated semantic and contextual information into the tensor space model using contextual embedding vectors and proposed a 2-channel tensor model that effectively handles sequential information as input. The contextual embedding-based tensor space model includes a term axis composed of words that help in text classification. In the process of selecting the words that make up the term axis of the tensor space model, these words lose their sequence information. In this paper, we proposed a 2-channel deep learning architecture that adds a *sequence channel* to manage sequence information, thereby enhancing the training performance of tensor space models that previously did not include sequence information. The 2-channel tensor deep learning architecture was devised to effectively process semantic information, context information, and sequence information by separately training the ELMo (or BERT) channel composed of contextual embedding vectors and the *Transformer*-based *sequence channel*. Using six English text datasets, we experimentally demonstrated that the proposed method performs better than existing models. Furthermore, we described the process of optimizing the hyperparameters of the proposed model and provided a table of optimal hyperparameters to enable readers to implement our model; this allows researchers to realize not only sentiment analysis and fake news detection, but also for various text classification tasks. We hope that the performance improvement demonstrated in this study will contribute to performance enhancements in real-world applications. In addition to semantics, context, and sequence information, there are other factors, such as keywords and tag information, that can be considered for text classification. Future work includes another multi-channel architecture with new channels for learning this additional meta-information.

## REFERENCES

1. Da'u A., Salim N., "Aspect Extraction on User Textual Reviews Using Multi-Channel Convolutional Neural Network," *PeerJ Computer Science*, Vol. 5, 2019, e191.
2. Guo B., Zhang C., Liu J., "Improving text classification with weighted word embeddings via a multi-channel TextCNN model," *Neurocomputing* Vol. 363, 2019, pp. 366-374.
3. Peng H., Xu L., Bing L., Huang F., Lu W., Si L., "Knowing What, How and Why: A Near Complete Solution for Aspect-Based Sentiment Analysis," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 8600-8607.
4. Maltoudoglou L., Paisios A., Lenc L., Martínek J., Král P., Papadopoulos H., "Well-Calibrated Confidence Measures for Multi-Label Text Classification with a Large Number of Labels," *Pattern Recognition*, Vol. 122, 2022, 108271.
5. Lin J.W., Thanh T.D., Chang R.G., "Multi-Channel Word Embeddings for Sentiment Analysis," *Soft computing*, Vol. 26, 2022, pp. 12703-12715.
6. Mueller A., Krone J., Romeo S., Mansour S., Mansimov E., Zhang Y., Roth D., "Label Semantic Aware Pre-Training for Few-Shot Text Classification," in

*Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2022, pp. 8318-8334.
7. Bird J., Ekárt A., Faria D.R., "Chatbot Interaction with Artificial Intelligence: human data augmentation with T5 and language transformer ensemble for text classification," *Journal of Ambient Intelligence and Humanized Computing*, Vol. 14, 2023, pp. 3129-3144.
8. Peters M.E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., Zettlemoyer L., "Deep Contextualized Word Representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1, 2018, pp. 2227-2237.
9. Devlin J., Chang M.W., Lee K., Toutanova K., "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1, 2019, pp. 4171-4186.
10. Kim H.J., Chang J.Y., "A Semantic Text Model with Wikipedia-Based Concept Space," *Journal of Society for e-Business Studies*, Vol 19, 2014, pp. 107-123.
11. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Polosukhin I., "Attention is all you need," *Advances in Neural Information Processing Systems*, Vol. 30, 2017, pp. 5998-6008
12. Mikolov T., Chen K., Corrado G., Dean, J., "Efficient Estimation of Word Representations in Vector Space," *arXiv preprint arXiv:1301.3781*, 2013.
13. Pennington J., Socher R., Manning C.D., "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532-1543.
14. Ricardo B.Y., Berthier R.N., "Modern Information Retrieval: The Concepts and Technology behind Search, 2/E," 2011, pp. 62-77.
15. Sutskever I., Vinyals O., Le Q. V., "Sequence to Sequence Learning with Neural Networks," *Advances in neural information processing systems*, Vol. 27, 2014.
16. Bahdanau D., Cho K.H., Bengio Y., "Neural Machine Translation by Jointly Learning to Align and Translate," *arXiv preprint arXiv:1409.0473*, 2014.
17. Xie J., Hou Y., Wang Y., Wang Q., Li B., Lv S., Vorotnitsky Y. I., "Chinese Text Classification Based on Attention Mechanism and Feature-Enhanced Fusion Neural Network," *Computing*, Vol. 102, 2020, pp. 683-700.
18. Chrysostomou G., Aletras N., "Improving the Faithfulness of Attention-based Explanations with Task-specific Information for Text Classification," *arXiv preprint arXiv:2105.02657*, 2021.
19. Chang W.C., Yu H.F., Kai A., Amazon Z., Yang Y., Dhillon I.S., Zhong K., "Taming Pretrained Transformers for Extreme Multi-Label Text Classification," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 3163-3171.
20. Kano T., Sakti S., Nakamura S., "Transformer-based direct speech-to-speech translation with transcoder," in *IEEE Spoken Language Technology Workshop*, 2021, pp. 958-965.
21. Yang J., Gupta A., Upadhyay S., He L., Goel R., Paul S., "TABLEFORMER: Robust Transformer Modeling for Table-Text Encoding," in *Proceedings of the*

*Annual Meeting of the Association for Computational Linguistics*, 2022, pp. 528-537.
22. Yu G., Li A., Zheng C., Guo Y., Wang Y., Wang H., "Dual-Branch Attention-in-Attention Transformer for Single-Channel Speech Enhancement," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2022, pp. 7847-7851.
23. Kim Y., "Convolutional Neural Networks for Sentence Classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746-1751.
24. Yao L., Mao C., Luo Y., "Graph Convolutional Networks for Text Classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, no. 01, 2019, pp. 7370-7377.
25. Lin Y., Meng Y., Sun X., Han Q., Kuang K., Li J., Wu F., "BertGCN: Transductive Text Classification by Combining GNN and BERT," *Findings of the Association for Computational Linguistics* (ACL-IJCNLP 2021), 2021, pp. 1456–1462.
26. Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L., Stoyanov V., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv prepresnt arXiv.1907.11692*, 2019.
27. He P., Liu X., Gao J., Chen W., "DeBERTa: Decoding-enhanced BERT with Disentangled Attention, " *arXiv preprint arXiv.2006.03654*, 2021.
28. Zhao W.X., Zhou K., Li J., Tang T., Wang X., Hou Y., Min Y., Zhang B., Zhang J., Dong Z., Du Y., Yang C., Chen Y., Chen Z., Jiang J., Ren R., Li Y., Tang X., Liu Z., Liu P., Nie J.Y., Wen J.R., "A Survey of Large Language Models," *arXiv preprint arXiv.2303.18223*, 2023.
29. Sun X., Li X., Li J., Wu F., Guo S., Zhang T., Wang G., "Text Classification via Large Language Models," arXiv preprent arXiv.2305.08377, 2023.
30. Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., Iwasawa, Y., "Large language models are zero-shot reasoners," Advances in Neural Information Processing Systems, Vol. 35, 2022, pp. 22199-22213.

**Han-joon Kim (**金漢峻**)** received the B.S. and M.S. degrees in Computer Science and Statistics from Seoul National University, Seoul, Korea in 1994 and 1996 and the Ph.D. degree in Computer Science and Engineering from Seoul National University, Seoul, Korea in 2002, respectively. He is currently a professor at the School of Electrical and Computer Engineering, University of Seoul, Korea. His current research interests include text mining, deep learning, big data analysis, and intelligent information retrieval.

**Soon Kwan Kwon (權純寬)** received the B.S. in Mathematics with a double major in Applied Statistics from Hoseo University, Korea in 2022 and M.S. degrees in Electronic and Computer Engineering from the University of Seoul, Korea. His research interests include text mining, deep learning, and big data analysis.