IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 13 June 2022, accepted 26 June 2022, date of publication 28 July 2022, date of current version 2 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3194650

RESEARCH ARTICLE

Spectral-Spatial Classification of Hyperspectral Images Using BERT-Based Methods With HyperSLIC Segment Embeddings

IBRAHIM ONUR SIGIRCI AND GOKHAN BILGIN
Department of Computer Engineering, Yildiz Technical University (YTU), Davutpasa Campus, 34220 Istanbul, Turkey
Signal and Image Processing Laboratory (SIMPLAB), Yildiz Technical University (YTU), 34220 Istanbul, Turkey

Corresponding author: Gokhan Bilgin (gbilgin@yildiz.edu.tr)

This work was supported by Yildiz Technical University, Scientific Research Projects Coordination Unit, under Project 2014–04-01–KAP01.

ABSTRACT The classification performance is highly affected because hyperspectral images include many bands, have high dimensions, and have few labeled training samples. This challenge is reduced by using rich spatial information and an effective classifier. The classifiers in this study are BERT-based (Bidirectional Encoder Representations from Transformers) models, which have recently been applied in natural language processing. The BERT model and its performance-improved version, the ALBERT (A Lite BERT) model, are utilized as transformer-based models. Because of their structure, these models can also accept spatial information via 'segment embeddings'. Segmentation algorithms are commonly used in the literature to get spatial information. Superpixel methods have shown superior results in the segmentation literature due to the utility of working at the superpixel level rather than the conventional pixel level. HyperSLIC, a modified version of the SLIC superpixel method for hyperspectral images, is employed as input to BERT-based models in this study. In addition, HyperSLIC segmentation results are merged with the DBSCAN algorithm for similar superpixels to increase the size of spatially similar areas and called as HyperSLIC-DBSCAN. The effects of segment embedding information on classification accuracy in BERT-based models is studied experimentally. Experimental results show that BERT-based models outperform conventional and deep learning-based 1D/2D convolutional neural network classifiers when spatial information is used with the help of segment embedding information.

INDEX TERMS ALBERT, BERT, classification, hyperspectral images, hyperSLIC, segmentation, transformers.

I. INTRODUCTION

Hyperspectral images are obtained through sensors that can capture the reflection of electromagnetic waves in the visible and infrared regions of the electromagnetic spectrum. It contains rich information both spatially and spectrally, as it consists of bands received at frequent intervals over a wide area of the spectrum. Therefore, it is used in many fields, such as earth sciences, astronomy, medical imaging, agriculture, food security, surveillance in defense and civil applications, target detection, land use, and urban planning. The values of the pixels in the same location in all bands create a meaningful signal specific to the substances in the area called the spectral signature. Spectral signatures containing different values for each object/material provide a discriminative input for classification and segmentation applications in terms of pattern recognition and machine learning. For this reason, it is basically used as an input feature in machine learning problems.

In the early stages of research in this area, only spectral signatures have been used as a feature in hyperspectral image classification tasks [1]–[3]. However, differences in

The associate editor coordinating the review of this manuscript and approving it for publication was Kumaradevan Punithakumar.

spectral signatures occur due to atmospheric effects, different angles of sunlight to the ground, and the presence of various substances/objects in a pixel region in hyperspectral images. In addition, the scarcity of data in this area makes it difficult to conduct adequate comparative experiments, while insufficient training data reduces the classification performances in the experiments. Studies conducted in those years showed that support vector machines (SVM) using radial basis kernel (RBF) gave better results among classifiers [4], [5]. In the following years of the literature, since spectral information did not give a satisfactory performance, spatial information began to be used effectively in classification [6]–[8]. In various studies, spatial information has been included in the system with different approaches in feature extraction, modeling, and post-processing stages, resulting in a significant increase in classification performance.

Tarabalka *et al.* have made essential studies on the use of both spectral and spatial information together in hyperspectral image classification. In [9], a single gradient was obtained using the robust color morphological gradient method with spectral signatures for the use of spatial information [10]. Then, a watershed segmentation algorithm was applied to the gradient and a segmentation map was obtained. The obtained segmentation map and pixel-based SVM classification map were combined with the majority voting method. In addition to [9] in [11], the concept of reliable point (marker) has been defined and included in the system. The reliable point information is the pixels selected on the probability map obtained at the output of the SVM classification algorithm. In [12], reliable pixel neighborhoods were classified using the minimum spanning tree method [13], and results were combined with the classification map by the majority voting method. The method proposed by Kazanskiy *et al.* [14] have combined the K-means++ segmentation map with the pixel-based SVM classification map using majority voting. Zhang *et al.* [15] have added the spatial information to the K-means algorithm in their proposed method. Thus, they have made a classification with the unsupervised learning method.

As in most image classification studies, the use of appropriate segmentation approaches to obtain spatial information in the classification of hyperspectral images seems to increase the classification performance. Superpixels, which contain more spatial information than fixed-size pixel regions, represent small areas that can be adapted to the size and shape of objects in the hyperspectral scene. Superpixel-based approaches used for grouping pixels that share a common signal characteristics were employed to generate segmentation maps [16], [17]. Bai *et al.* [18] used the adaptive graph structure method while using the SLIC superpixel algorithm for the required segmentation information. To include spatial information into their system, they used the attention matrix created by using the adjacency matrix of superpixels. Zhao *et al.* [19], developed a graph-based method based on superpixel maps of varying sizes to resolve the challenge of low accuracy and limited labeled data. Xie *et al.* [20] included superpixel techniques into the convolutional neural network they used for hyperspectral image classification. Pooling approaches on the network are built utilizing superpixel segmentation to solve the challenge of insufficient labeled training data.

During the development of neural network approaches as well as traditional machine learning approaches, extreme learning machines have been used for hyperspectral image classification [21], [22]. Many studies have been carried out using deep learning networks, which have gained considerable popularity in parallel with the development of fast processors and graphics processing units as hardware. In terms of hyperspectral image classification, many deep learning frameworks have been developed for spectral features, spatial features, and spatial-spectral features together [23], [24]. One- [25]–[27], two- [28] and three- [24] dimensional convolutional neural network (CNN) architectures are generally used in the hyperspectral studies. Chen *et al.* have used 3-D CNN-based feature model with combined regularization to extract effective spectral-spatial features [29]. Deep transfer learning aims to reuse a machine learning model trained for a similar problem by changing some weights of the model without changing the model architecture much, and is an important study area and approach for complex data such as hyperspectral images [30]. In the field of deep learning, many architectures such as GAN (Generative Adversarial Networks) [31], [32] and RNN (Recurrent Neural Networks) [33], [34] continue to be used by researchers.

The fact that spectral signatures are one-dimensional data and are treated as words related to materials has led to the evaluation of natural language processing algorithms in hyperspectral image classification [35]. In recent years, there have been significant developments in natural language processing with the deep learning approach. One of the most important of these advances is presenting the transformer-based BERT (Bidirectional Encoder Representations from Transformers) model developed to learn the semantic relationships between word tokens by Google to the literature [36]. He *et al.* have used the BERT model for the first time in hyperspectral images [37]. In the BERT model of their study, pixels and location information in specific windows were used as input and performance evaluation was made by training them according to different parameters. In the following months, Google has improved the BERT model and introduced the ALBERT (A Lite BERT) model to the literature [38]. Especially in the field of natural language processing, the ALBERT model showed higher performance than the BERT model. In addition, BERT's basic model consists of 108 million parameters, while ALBERT's basic model consists of 12 million parameters. Thus, while the ALBERT model takes up less memory, the training/testing processes are made faster at the same time. Recently, transformer-based approaches have been used in hyperspectral image classification in addition to the BERT model. Hong *et al.* [39] has used transformer structures as a backbone in their deep learning model, which they named SpectralFormer. In their work, they used input embeddings and position embeddings; but they did not use the 'segment embeddings' structure. He *et al.* [40] has utilized transformer encoders without 'segment embeddings' in the convolutional neural network-based method they proposed in their study.

In this study, the SLIC (Simple Linear Iterative Clustering) super pixel algorithm [41], which has a fast and comprehensible structure, is used in order to increase the classification performance of BERT models. In this study, the SLIC algorithm is modified to take into account spatial information in hyperspectral images. The modified approach called HyperSLIC is one of the novelties of the study. In addition to the BERT model in the literature, the ALBERT model is used to classify hyperspectral images by using spectral and spatial information together. Another innovative aspect of the study is that BERT and ALBERT models can use spectral/spatial information together by giving HyperSLIC superpixel information to the model as a segment embedding vector.

The remainder of this paper is structured as follows. Section II describes the SLIC superpixel algorithm and introduces the proposed HyperSLIC for hyperspectral images. Section III explains the structure of the BERT-based models. The proposed framework for the classification of hyperspectral images using HyperSLIC and the BERT-based models is described with using figures and flowchart in Section IV. Hyperspectral datasets and experimental results are presented in Section V. Section VI concludes this study with the final remarks and points out future work.

II. HyperSLIC

The Simple Linear Iterative Clustering (SLIC) algorithm, which performs a local clustering of pixels, is a k-means based algorithm that works with a defined distance metric using the CIELab color space and the x, y pixel position [41]. Although the algorithm is simple, it offers a very efficient approach to image segmentation. The first difference between the SLIC than k-means is that the distance calculations are limited by the superpixel size. Therefore, complexity of the method depends on the number of pixels and the number of superpixels. The second difference is that color and spatial proximity features are combined to control the size and compactness of superpixels.

Algorithm 1 shows the steps of the SLIC algorithm. The k parameter represents the approximate number of superpixels desired to be obtained from all the pixel counts (N) in the scene. Initially, pixel samples are selected from regions of dimensions $S \times S$ for k superpixels. The cluster centers (C_k) determined with the help of the lowest gradient in the window created in size of 3×3 within each region are updated. Then, the distance between C_k and each pixel in the $2S \times 2S$ region around cluster center is determined. When calculating the distance between each pixel and all cluster centers in the k-means algorithm; in the SLIC method, only the distances between the pixels in the $2S \times 2S$ region and the cluster centers are calculated. Next, each pixel is assigned to the nearest cluster center. The difference between the previous cluster centers and the new cluster centers is calculated as an error and the process continues until the error values found reach a certain threshold value..

Algorithm 1 Superpixel Segmentation With SLIC

```
 1: $S \leftarrow \sqrt{\frac{N}{k}}$  and  $l_t \leftarrow -1$  and  distance$_t \leftarrow \infty$
 2: $C_k \leftarrow [x_k, y_k, l_k, a_k, b_k]$ by sampling pixels at each grid steps $S$
 3: Cluster centers are moved to the lowest gradient position in a $3 \times 3$ neighborhood.
 4: repeat
 5:     for each $C_k$ do
 6:         for each pixel $t$ in $2S \times 2S$ region with center $C_k$ do
 7:             Dist $\leftarrow$ calculateDistance($C_k, t, S, m$)
 8:             if Dist $<$ distance$_t$ then
 9:                 distance$_t \leftarrow$ Dist
10:                 $l_t \leftarrow k$
11:     Re-computation cluster centers $C_k$
12:     error $\leftarrow \sum_k ||C_k^{\text{new}} - C_k^{\text{old}}||_{L2}$
13: until error $\leq$ threshold
```

Since superpixels contain pixel values and position information, distance calculations are made taking this information into account. The Equation (1) and Equation (2) show the distance measurement between two coordinates (dist$_s$) and the distance between two pixel values (dist$_c$), respectively.

$$\text{dist}_s \leftarrow \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{1}$$

$$\text{dist}_c \leftarrow \sqrt{(l_i - l_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2} \tag{2}$$

The calculated distance metrics dist$_s$ and dist$_c$ are normalized to eliminate the negative influence of each other. While the parameter S is used to normalize dist$_s$, the experimentally selected m value between $[1, 10]$ is used for dist$_c$. These distance metrics can be combined to get a single distance measure as follows:

$$\text{dist} \leftarrow \sqrt{(\frac{\text{dist}_c}{m})^2 + (\frac{\text{dist}_s}{S})^2} \equiv \sqrt{\text{dist}_c^2 + (\frac{\text{dist}_s}{S} \cdot m)^2} \tag{3}$$

The values of a pixel/spectral signature in a hyperspectral image can be represented as $p_i = [p_{i1}, p_{i2}, \ldots, p_{id}]$, where d indicates the number of spectral bands. According to this, Equation (2) can be changed as follows:

$$\text{dist}_c \leftarrow \sqrt{(p_{i1} - p_{j1})^2 + (p_{j2} - p_{j2})^2 + \cdots + (p_{id} - p_{jd})^2} \tag{4}$$

The coordinate distance dist$_s$ is calculated with the same formula and the S value can be used for normalization. In this situation, m parameter should be determined. If $1 \leq k \leq d$ and $0 \leq p_{ik} \leq 255$ so $0 \leq (p_{ik} - p_{jk})^2 \leq 255^2$, the value of m can be determined as follows:

$$0 \leq \text{dist}_c \leq 255\sqrt{d} = m \tag{5}$$

For adaptation to hyperspectral images, Equation (3) can be updated as in Equation (6). In this study, the distance

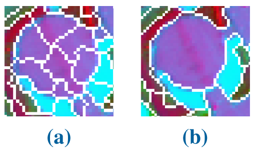

FIGURE 1. An example of merging superpixels by DBSCAN; (a) SLIC result and (b) SLIC-DBSCAN result.

metric in the SLIC method is changed and named as the HyperSLIC metric as follows:

$$\text{dist} \leftarrow \sqrt{\text{dist}_c^2 + \text{dist}_s^2 \cdot (\frac{255\sqrt{d}}{S})^2} \tag{6}$$

A. HyperSLIC-DBSCAN

In Kovesi's study [42], superpixels obtained with SLIC are clustered with the DBSCAN algorithm [43]. Thus, superpixels similar to each other according to the Euclidean metric are merged to form large segments. Algorithm 2 shows the DBSCAN steps of the SLIC-DBSCAN algorithm. According to the SLIC-DBSCAN algorithm, all superpixels are processed starting from any of the superpixels. If the inspected superpixel is previously included in a cluster, it is passed to the other superpixel without further processing. If the superpixel is not included in any cluster, a regionQuerySimilarity is used to find the superpixel neighborhood at ϵ distance. If the number of neighbors is greater than MinPTS, this superpixel and its neighbors are treated as a new cluster. With expandClustering, new neighbors are found by the new regionQuerySimilarity for each previously unclustered neighbor. Neighborhood finding is the most computationally demanding part of the DBSCAN algorithm. As an example, the merging of superpixels in Fig. 1(a) can be shown in Fig. 1(b).

Algorithm 2 Clustering of Superpixels Using DBSCAN

```
1:  $C \leftarrow 0$
2:  Visited$_{\text{all superpixel}}$ $\leftarrow$ 0, mark unvisited all superpixel
3:  for each unvisited superpixel s do
4:      Visited$_s$ = 1
5:      NeighborSuperpixels $\leftarrow$ regionQuerySimilarity($s$, $\epsilon$)
6:      if sizeof(NeighborSuperpixels) $<$ MinPts then
7:          Visited$_s$ $\leftarrow$ 2, mark noise
8:      else
9:          $C$ $\leftarrow$ next cluster
10:         expandCluster(P, NeighborSuperpixels, $C$, $\epsilon$, MinPts)
```

With a similar approach, the HyperSLIC segmentation result is also merged with DBSCAN and called the HyperSLIC-DBSCAN segmentation. As a novel inspired approach, the similarities between the superpixels in the HyperSLIC-DBSCAN algorithm are computed based on the universal image quality index as introduced in [44]. This similarity metric is named as spectral similarity index (sim_{ssi}). As in Algorithm 2, superpixels similar to the sim_{ssi} with a predefined ϵ threshold are found as follows Equation (7):

$$\text{sim}_{\text{ssi}} = \frac{4\sigma_{\mathbf{xy}}\bar{\mathbf{x}}\bar{\mathbf{y}}}{(\sigma_{\mathbf{x}}^2+\sigma_{\mathbf{y}}^2)[\bar{\mathbf{x}}^2+\bar{\mathbf{y}}^2]} = \frac{\sigma_{\mathbf{xy}}}{\sigma_{\mathbf{x}}\sigma_{\mathbf{y}}} \cdot \frac{2\bar{\mathbf{x}}\bar{\mathbf{y}}}{\bar{\mathbf{x}}^2+\bar{\mathbf{y}}^2} \cdot \frac{2\sigma_{\mathbf{x}}\sigma_{\mathbf{y}}}{\sigma_{\mathbf{x}}^2+\sigma_{\mathbf{y}}^2} \tag{7}$$

where $\mathbf{x}$ and $\mathbf{y}$ show representative/mean pixels of a superpixel patch. $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ show the averages of representative pixels, $\sigma_{\mathbf{y}}^2$ and $\sigma_{\mathbf{x}}^2$ show the variance of representative pixels and $\sigma_{\mathbf{xy}}$ show the covariance of pixels.

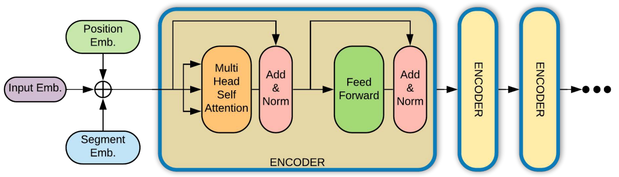

FIGURE 2. Overall structure of the BERT model architecture.

III. BERT-BASED MODELS

Google introduced the encoder-based BERT [36] model to the literature in November 2018. With its high performance on the SQuAD [45] dataset, it has shown that an important milestone has been reached in the field of natural language processing. Since the model includes many parameters due to its architectural structure, the training process takes a long time. However, it provides a significant performance boost in understanding the context in the text. BERT language model is constructed by training in two tasks: next sentence prediction (NSP) and masked language prediction (MLP). In the NSP task, the model is trained on whether two sentences come together semantically. In the MLP task, the model is trained to predict the masked tokens correctly.

Fig. 2 shows the overall structure of the BERT model. The BERT has 12 encoders in the base model and 24 encoders in the large model. The BERT model takes the following information as input: (i) the embedding vectors corresponding to the word tokens, (ii) the position embedding vectors for the positions of the word in the sentence and, (iii) the segment embedding vectors using the order in which the sentence containing the word appears in the input. These vectors are summed and given as input to the encoder. Encoders can be added consecutively because the input and output data of the encoders are the same size. After the last encoder, fully connected layers are added, then the weights are updated by calculating the losses.

In 2019, Google introduced the ALBERT [38] model to the literature and eliminated the disadvantages of the BERT model, such as high memory space and consuming too much time for the language model training. There are three differences between ALBERT and BERT models:

1) Sentence Order Prediction (SOP): Implementation of SOP task instead of NSP in language model training.
2) Matrix factorization: Using the product of two smaller matrices for input embedding vectors instead of a single large matrix.

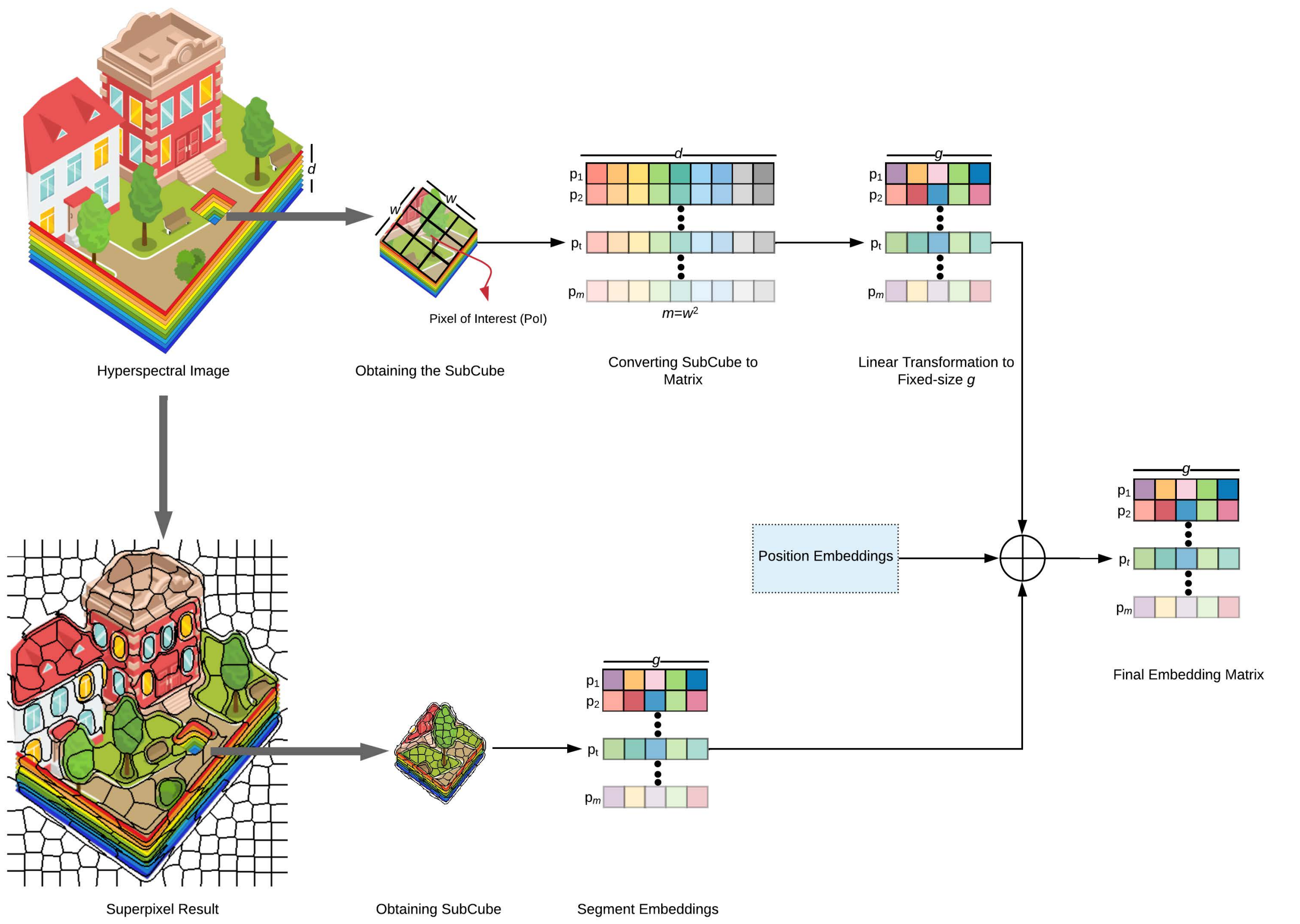

FIGURE 3. The flowchart of the input format formation for the classification by using the hyperspectral images and the segmentation results obtained by the superpixel method together.

3) Parameter Sharing: Sharing parameters between encoders.

In the ALBERT, the BERT model parameters have been reduced from 108 million to 12 million using the matrix factorization and parameter sharing arrangements. With the SOP, the context understanding of the model is also increased. Thus, the ALBERT model, which is faster, using less memory, and understands the context, is presented to the literature. A detailed review of the ALBERT model and transition approaches from natural language processing to hyperspectral image processing are described in the proposed approach (Section IV).

IV. THE PROPOSED METHOD

In order to understand the overall proposed system structure, the stages of input format formation for classification by using the hyperspectral images and the segmentation results obtained by the superpixel method together for ALBERT model are shown in Fig. 3. The complementary flowchart for the classification of the final embeddins of a '*pixel of interest*' with the BERT-based architecture is given in Fig. 4.

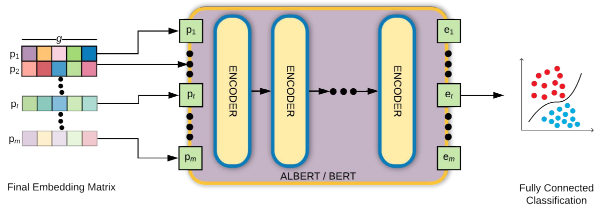

FIGURE 4. The complementary flowchart for the classification of the final embeddins of a '*pixel of interest*' with the BERT-based architecture.

Please note that, the term '*pixel of interest*' is used for training and testing pixels together.

Fig. 3 and 4 depict the proposed technique. Under the proposed method section, Fig. 3 is describe in A and B, whereas Fig. 4 is describe in C and D.

A. OBTAINING THE SubCube OF A PIXEL OF INTEREST AND FLATTENING

For each pixel of interest in the hyperspectral image cube, a small 'SubCube' is created by windowing around a pixel,

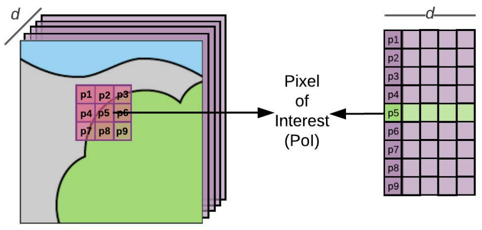

FIGURE 5. 'Pixel of Interest' neighborhood relations and flattening phase in SubCube structure.

containing its neighbors at a certain distance. So, the dimension of each SubCube is $w \times w \times d$, where d indicates the number of spectral bands of hyperspectral image and w shows the dimension of the window taken around the center pixel as in Fig. 3. Then the SubCubes are flattened to be a matrix of dimensions $m \times d$ as shown in Fig. 5, where $m = w^2$ and the 'pixel of interest' is set to be in the middle row of the matrix.

B. EMBEDDING VECTORS

BERT-based models use the sum of three different embedding vectors: *i*) token embeddings, *ii*) position embeddings and *iii*) segment embeddings.

1) TOKEN EMBEDDINGS

Token embeddings are constant-size vectors expressing numerical representations of word tokens. In this study, spectral signatures correspond to these vectors; however, the size of spectral signatures differs according to hyperspectral datasets. Also, the size of the vector lengths is limited due to the architecture of the encoders. This problem can be solved by making the spectral signatures a fixed-size using a linear transformation. For transformation, signatures of d length are multiplied by the W transformation matrix to become signatures of g length. Since the W matrix is designed as part of the BERT-based models, its values are learned during the training.

2) POSITION EMBEDDINGS

Position embeddings are fixed-size vectors that represent the order of word tokens in the input. In this study, the location of the pixel in the matrix (from 1 to m) after flattening is taken as the position embedding.

3) SEGMENT EMBEDDINGS

In NLP studies, two sentences are entered into to the ALBERT model during language model training. The vector representations that show which sentence the word tokens belongs to are called segment embedding. In this part, which is one of the novelties in this study, segmentation maps are used as segment embedding, showing which segment the pixels belongs to. Segmentation maps are obtained using the HyperSLIC method, which is another novelty proposed in this study. As the output of HyperSLIC method, small and compatible segments are obtained and the representation

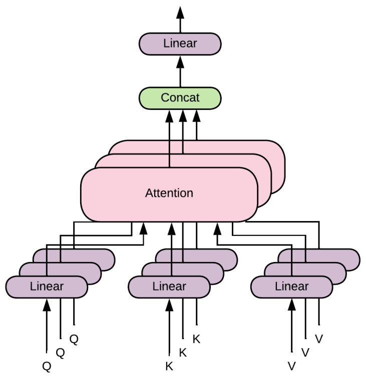

FIGURE 6. Multihead self-attention architecture (self-attention when Q, K, V have the same values).

ability of these segments is higher than other segmentation algorithms. Thus, the vector representation corresponding to each segment is used as the segment embedding input.

However, having a large number of superpixels increases the number of vector representations. This makes it difficult to learn similar superpixels with different vector representations. As a solution to this, adjacent superpixels that are similar to each other are combined in the HyperSLIC-DBSCAN algorithm and learning is performed with fewer segments. The vector representation of all pixels in a segment is the same, and this information is given as input to the BERT-based models for every pixel.

C. ENCODER

BERT-based models are based on encoders with the same size of inputs and outputs. When parameter sharing is used, the same variable is used in each encoder. This feature represents another novelty of ALBERT over the BERT model. The structure of encoders is designed to learn the relationship between embedding vectors. The most important part of its structure is the 'Multihead Self Attention (MHSA)'.

1) MULTIHEAD SELF ATTENTION (MHSA)

The MHSA structure, shown in Fig. 6, employed in the encoders is one of the important building blocks of BERT-based models. This structure performs mathematical operations to learn the relationships between embedding vectors that are encoder inputs. Here, **Q**, **K**, **V** matrices represent the input matrices to the encoder. If all three of these matrices are the same, it is called *self-attention*. These matrices are transformed linearly with weights to mathematically represent the relationship between word tokens (pixels in this study). This structure is used multiple times in the BERT-based models. The *self-attention* structure can be expressed by Equation (8):

$$\begin{aligned} h_i &= Attention(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V) \\ &= Attention(\mathbf{XW}_i^Q, \mathbf{XW}_i^K, \mathbf{XW}_i^V) \end{aligned} \tag{8}$$

where, h_i is the result of the attention structure of i. The $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ show the weight matrices used in linear transformations. These matrices are multiplied by the encoder input **X**. The **Q**, **K**, **V** matrices are equal to **X** matrix given to encoder as input. In other words, since the three matrices are the same, a *self-attention* structure is obtained.

2) FEED FORWARD

With the help of this module, linear transformations are performed for the next encoder task. The feed-forward module consists of two linear transformation layers. The ReLU (Rectified Linear Unit) activation function is used in each layer.

3) LAYER NORMALIZATION

Layer normalization has beeb proposed by Ba *et al.* [46]. With this module, the contribution of features during training is normalized. In batch normalization method, normalization is applied to layers instead of batches.

D. CLASSIFICATION LAYER

BERT-based models generate new vectors based on the relationship of the 'pixel of interest' to its neighbors. The vector corresponding to the 'pixel of interest' (the vector in the middle row of the matrix) is given as an input to the three-level fully-connected layer. It is structured so that the number of neurons in the last layer is equal to the number of classes in the dataset. This layer added to classify the input vector is called the classification layer. The output of this layer is used to calculate the error/loss values and update the parameters of the model.

In the classification layer, there is a fully connected network. Due to its network structure, it produces output as much as the number of classes. The predicted and actual outputs are used to calculate errors. The error is calculated using categorical cross-entropy. The Adam algorithm [47] is used as the optimizer, with 'learning rate=$2e-5$' and 'epsilon=$1e-8$'.

V. EXPERIMENTAL RESULTS

In the experiments, the contribution of the number of encoders used, k window size, number of training data, position embeddings, segment embeddings for the BERT-based models are investigated and these contributions are presented compararatively. All of the computational work is done on Intel(R) Core(TM) i7-6700K 4.00GHz CPU with 64GB RAM. Python Keras [48], TensorFlow [49], sklearn [50] libraries are used for the experiments.

A. HYPERSPECTRAL DATASETS

Pavia University and Indian Pines datasets, which are used extensively in the field of hyperspectral image processing, are used to evaluate the proposed methods.

Pavia University dataset was captured with the ROSIS-03 (Reflective Optics System Image Spectrometer) optical sensor on the University of Pavia, Italy by Deutschen Zentrum für Luftund Raumfahrt (DLR, German Aerospace Agency). The hyperspectral scene has 115 spectral bands between $0.43\mu m - 0.86\mu m$ spectral coverage. Twelve bands were removed due to high noise and were presented as 103 bands. The dataset consists of 610 rows and 340 column pixels, and each pixel has 1.3 meters spatial resolution. The dataset contains nine different classes: asphalt, meadow, gravel, tree, metal sheet, bare soil, bitumen, brick, and shadow. There are 42776 labeled pixels labeled in the released dataset. Fig. 7 shows the false-color image of Pavia University dataset

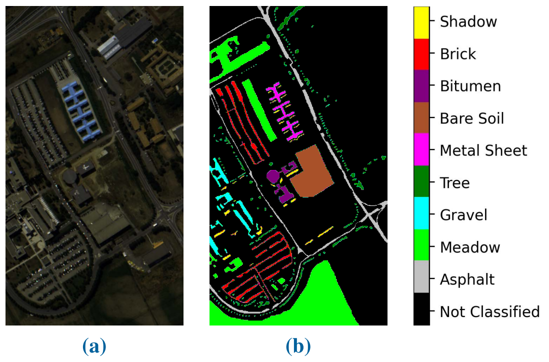

FIGURE 7. Pavia University dataset (a) false-color image and (b) ground-truth data.

TABLE 1. Class names and the number of the labeled training and test pixels for Pavia University dataset.

Class Number	Class Name	Total Samples	Train Samples	Test Samples
1	Asphalt	6631	150	6481
2	Meadow	18649	150	18499
3	Gravel	2099	150	1949
4	Tree	3064	150	2914
5	Metal Sheet	1345	150	1195
6	Bare Soil	5029	150	4879
7	Bitumen	1330	150	1180
8	Brick	3682	150	3532
9	Shadow	947	150	797
	Total	**42776**	**1350**	**41426**

TABLE 2. Class names and the number of the labeled training and test pixels for Indian Pines dataset.

Class Number	Class Name	Total Samples	Train Samples	Test Samples
1	Alfalfa	46	23	23
2	Corn-no till	1428	150	1278
3	Corn-min till	830	150	680
4	Corn	237	118	119
5	Grass/pasture	483	150	333
6	Grass/trees	730	150	580
7	Grass/ pasture-mowed	28	14	14
8	Hay-windrowed	478	150	328
9	Oats	20	10	10
10	Soybeans-no till	972	150	822
11	Soybeans-min till	2455	150	2305
12	Soybeans-clean till	593	150	443
13	Wheat	205	102	103
14	Woods	1265	150	1115
15	Bldg-grass-tree-drives	386	150	236
16	Stone-steel towers	93	46	47
	Total	**10249**	**1813**	**8436**

TABLE 3. Performance comparisons with conventional, deep learning, BERT-based (*without segment embeddings*) classifiers on Pavia University dataset.

Classifier Type	Methods	Evaluation Results (%)				
Conventional Classifiers	kNN	71.45				
	SVM	72.14				
	RF	68.97				
DL-based Classifiers	1D CNN	74.87				
	2D CNN	86.22				
BERT-based Classifiers (without SE)		**Number of Encoders**	**Window Sizes**			
			3×3	**5×5**	**7×7**	**9×9**
	ALBERT	1	68.77	72.57	77.44	84.83
		3	72.60	76.54	81.14	85.13
		5	78.27	79.39	82.52	90.77
		9	74.92	80.89	87.27	86.52
		12	80.99	81.79	89.07	89.15
	BERT	1	70.76	73.01	78.00	82.20
		3	71.59	83.06	87.07	88.76
		5	73.75	85.21	87.16	89.21
		9	79.64	86.49	90.31	93.94
		12	78.47	87.13	88.71	88.86

TABLE 4. Performance comparisons with conventional, deep learning, BERT-based (*without segment embeddings*) classifiers on Indian Pines dataset.

Classifier Type	Methods	Evaluation Results (%)				
Conventional Classifiers	kNN	59.42				
	SVM	74.63				
	RF	69.12				
DL-based Classifiers	1D CNN	72.04				
	2D CNN	76.02				
BERT-based Classifiers (without SE)		**Number of Encoders**	**Window Sizes**			
			3×3	**5×5**	**7×7**	**9×9**
	ALBERT	1	74.22	76.39	80.28	81.08
		3	77.26	83.24	85.63	87.53
		5	81.06	84.06	88.50	87.07
		9	82.85	84.86	84.28	89.41
		12	78.84	87.81	83.23	86.82
	BERT	1	71.80	76.94	79.6	83.85
		3	80.69	87.45	88.24	89.85
		5	84.77	89.31	89.86	90.58
		9	88.21	89.41	91.37	91.83
		12	88.29	89.59	93.84	93.46

and ground-truth. Table 1 contains information about the classes, the number of labeled training and test pixels in the dataset [51].

Indian Pines hyperspectral dataset was captured with AVIRIS (Airborne Visible / Infrared Imaging Spectrometer) optical sensor on Northwest Indiana, USA and widely used in hyperspectral studies. The hyperspectral scene 224 spectral bands between $0.4\mu m - 2.5\mu m$ spectral coverage. Due to water absorption, 24 bands were deleted from the data and the remaining 200 bands were used in the experiments. The dataset consists of 145 rows and 145 column pixels; each pixel has a spatial resolution of 20 meters. The dataset contains 16 different classes: alfalfa, corn-no till, corn-min till, corn, grass/pasture, grass/trees, grass/pasture-mowed, hay-windrowed, oats, soybeans-no till, soybeans-min till, soybean-clean, wheat, woods, bldg-grass-tree-drives, stone-steel towers. There are 10249 pixels labeled in the released dataset without splitting. Fig. 8 shows the false-color image of Indian Pines dataset and ground-truth. Table 2 shows information about the classes, the number of labeled training and test pixels in the dataset [51].

B. RESULTS OF STUDIES

In this study, the performance evaluation of transformer-based machine learning techniques, BERT and ALBERT methods,

TABLE 5. **Number of parameters according to number of encoders in BERT and ALBERT models.**

# of Encoders	ALBERT	BERT
1	8,601,609	8,601,609
3	8,601,609	22,777,353
5	8,601,609	36,953,097
9	8,601,609	65,304,585
12	8,601,609	86,568,201

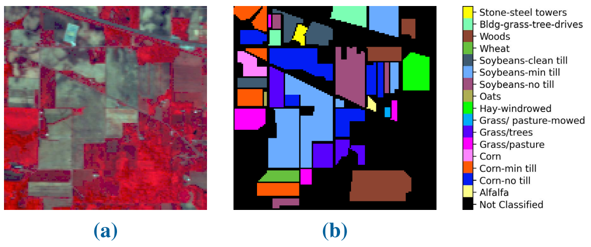

FIGURE 8. **Indian Pines dataset (a) false-color image and (b) ground-truth data.**

is proposed for the classification of hyperspectral images and the results are presented comparatively using quantitative performance metrics in two experimental studies:

1) EXPERIMENTAL STUDY-I

In this experiment, classification performances of ALBERT and BERT models are compared with conventional and deep learning-based (DL-based) classifiers. For conventional approaches, k-nearest neighbor (kNN) with k=9, Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel, Random Forest (RF) with 200 decision trees are employed to obtain optimized results. For deep learning-based approaches, 1D CNN (Convolutional Neural Network) [52] and 2D CNN [24] are carried out using the methods in related studies. The results of ALBERT and BERT models without segment embeddings (SE) are presented comparatively in Table 3 for Pavia University dataset and in Table 4 Indian Pines dataset. In the comparisons, the SubCube window sizes are chosen as $w = 3, 5, 7, 9$ and the number of encoders are chosen as 1, 3, 5, 9, 12.

In the first experimental study, DL-based algorithms show higher performance than conventional algorithms. BERT-based models without segment embedding information are able to achieve better results than deep learning methods with the increasing number of encoders. When the ALBERT and BERT model results are compared, it is seen that BERT gives better results in small patch sizes, and ALBERT closes the gap as the patch size grows. According to the results in Table 3 and Table 4, it is observed that the ALBERT model, which is smaller in size compared to BERT, produces results that will be an alternative to BERT model.

The Table 5 shows the number of parameters according to number of encoders in BERT and ALBERT models. According to the Table 5, since the ALBERT model allows the parameters to be shared, the number of parameters does not increase as the number of encoders increases. It should be noted that when the number of encoders is 12 in the BERT model, the number of parameters increases ten times compared to ALBERT. This is the disadvantage of the BERT model compared to ALBERT, and it increases the training time and size. As shown in Fig. 3, the size of the final embedding matrix is affected by the window size. However, since ALBERT and BERT models are large-size models, increasing the window size adds only a few parameters to the total parameter count. As a result, instead of giving the number of parameters that change according to the window

TABLE 6. **Experimental results for Pavia University dataset: (a) ALBERT and (b) BERT model.**

Number of Encoders	(a) **ALBERT** *with Different SubCube Window Sizes* (w × w)											
	without SE				with SE (HyperSLIC)				with SE (HyperSLIC-DBCAN)			
	3 × 3	5 × 5	7 × 7	9 × 9	3 × 3	5 × 5	7 × 7	9 × 9	3 × 3	5 × 5	7 × 7	9 × 9
1	68.77	72.57	77.44	84.83	97.37	98.23	98.85	**99.04**	97.70	98.76	**99.24**	**99.49**
3	72.6	76.54	81.14	85.13	96.96	97.77	98.58	**99.03**	97.62	**99.18**	**99.27**	**99.48**
5	78.27	79.39	82.52	**90.77**	96.24	96.90	98.38	98.67	97.84	**99.07**	**99.36**	**99.54**
9	74.92	80.89	87.27	86.52	95.57	97.10	97.95	98.79	98.43	**99.01**	**99.47**	**99.50**
12	80.99	81.79	89.07	89.15	96.38	96.40	97.75	98.59	98.62	**99.03**	**99.23**	**99.42**

Number of Encoders	(b) **BERT** *with Different SubCube Window Sizes* (w × w)											
	without SE				with SE (HyperSLIC)				with SE (HyperSLIC-DBCAN)			
	3 × 3	5 × 5	7 × 7	9 × 9	3 × 3	5 × 5	7 × 7	9 × 9	3 × 3	5 × 5	7 × 7	9 × 9
1	70.76	73.01	78.00	82.20	97.48	98.19	**99.38**	**99.37**	97.83	**99.06**	**99.1**	**99.35**
3	71.59	83.06	87.07	88.76	96.49	98.21	98.41	**99.21**	98.05	**99.11**	**99.44**	**99.68**
5	73.75	85.21	87.16	89.21	95.21	98.38	98.28	98.76	97.88	**99.31**	**99.39**	**99.71**
9	79.64	86.49	90.31	**93.94**	96.45	97.47	98.22	**99.17**	98.41	**99.13**	**99.30**	**99.50**
12	78.47	87.13	88.71	88.86	95.40	97.32	97.78	98.98	97.92	**99.11**	**99.33**	**99.47**

TABLE 7. Experimental results for Indian Pines dataset: (a) ALBERT and (b) BERT model results.

Number of Encoders	**(a) ALBERT** *with Different SubCube Window Sizes* ($\mathbf{w} \times \mathbf{w}$)											
	without SE				**with SE (HyperSLIC)**				**with SE (HyperSLIC-DBCAN)**			
	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9
1	74.22	76.39	80.28	81.08	97.42	98.06	98.68	98.00	97.58	98.38	98.26	98.46
3	77.26	83.24	85.63	87.53	98.15	98.79	98.87	**99.02**	97.68	98.76	98.74	98.67
5	81.06	84.06	88.50	87.07	98.10	**99.20**	**99.03**	**99.02**	98.02	98.72	98.96	98.80
9	82.85	84.86	84.28	**89.41**	98.27	**99.31**	**99.17**	**99.09**	98.09	98.71	**99.03**	98.77
12	78.84	87.81	83.23	86.82	97.24	98.77	98.72	98.66	98.23	98.90	**99.03**	**99.04**

Number of Encoders	**(b) BERT** *with Different SubCube Window Sizes* ($\mathbf{w} \times \mathbf{w}$)											
	without SE				**with SE (HyperSLIC)**				**with SE (HyperSLIC-DBCAN)**			
	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9
1	71.80	76.94	79.6	83.85	97.88	98.55	98.62	98.35	97.65	98.21	98.2	98.41
3	80.69	87.45	88.24	89.85	97.83	98.73	98.89	98.90	97.70	98.86	98.87	98.65
5	84.77	89.31	89.86	90.58	98.02	98.92	**99.04**	98.98	97.98	98.84	98.84	98.87
9	88.21	89.41	91.37	91.83	97.96	98.99	**99.12**	98.93	97.98	98.89	**99.10**	98.77
12	88.29	89.59	**93.84**	93.46	98.15	**99.00**	**99.08**	**99.04**	98.07	**99.03**	**99.09**	98.87

size, the average number of parameters for each encoder is presented in Table 5. An important point to note here is that ALBERT and BERT models have a similar structure when the number of encoders is one. Therefore, when the number of encoders in the result tables is one, the results obtained are very close to each other.

2) EXPERIMENTAL STUDY-II

In this experiment, segment embedding information is given as input to ALBERT and BERT models by superpixel segments as a novel approach. In accordance with this purpose, it is proposed to use segment embedding information using segmentation maps obtained from HyperSLIC and HyperSLIC-DBSCAN methods. The SLIC algorithm, which produces superpixel segments, is modified to use spatial information in hyperspectral images and called as HyperSLIC. In addition, HyperSLIC segmentation results are merged with the DBSCAN algorithm for similar superpixels to increase the size of spatially similar areas and called as HyperSLIC-DBSCAN. In Table 6 and Table 7 the results are in three main columns: *i*) without segment embedding, *ii*) segment embedding with HyperSLIC method and *iii*) segment embedding with HyperSLIC-DBCAN presented comparatively. In the comparisons, the SubCube window sizes are chosen as $w = 3, 5, 7, 9$ and the number of encoders are chosen as 1, 3, 5, 9, 12.

Table 6 shows the results obtained for the Pavia University dataset. The results for the Indian Pines dataset are shown in Table 7. When the results are examined, it is seen that BERT-based models with segment embedding information show higher performance. Furthermore, the best performance has been obtained using HyperSLIC-DBSCAN, which merges similar superpixels. The classification accuracy increases as the number of encoders and window size grow in the ALBERT model. When the results in Tables 6 and 7 are analyzed, it can be observed that by including segment information in the models, ALBERT approaches BERT in terms of classification performance and becomes a model as convenient as BERT.

TABLE 8. The training times of the BERT and ALBERT models in seconds.

Model	Number of Encoders	Window Sizes			
		3×3	5×5	7×7	9×9
ALBERT	**1**	61.3490	61.8960	70.2936	78.9229
	3	63.9576	74.6000	82.2978	99.4119
	5	74.4665	87.1518	95.1250	110.8282
	9	93.6424	106.8354	124.7600	144.2895
	12	108.7457	125.9785	145.7421	168.0942
BERT	**1**	57.6171	68.7566	76.2690	89.7089
	3	96.3254	111.2403	123.1069	143.4226
	5	128.1130	144.9758	164.2984	190.5928
	9	176.2533	196.4983	224.5711	264.3342
	12	216.6586	240.2638	275.5349	326.4110

Table 8 shows the training times in seconds for the ALBERT and BERT models. Models use the sum of input, position and segment embeddings as inputs. The segment embeddings do not affect the training time since adding or removing segment embeddings from the input does not affect the input size. The training period increases as the number of encoders increases, as seen in Table 8. As the number of encoders increases, the ALBERT significantly outperforms the BERT model. The BERT model took twice as long to train as the ALBERT model when there were 12 encoders. Increasing the window size also increases the size of the input matrix, which increases the training time. Similarly, when the window size increases, so does the size of the input matrix and, therefore, the training time. The ALBERT model with a single encoder and a 9×9 window size takes

TABLE 9. Results of McNemar's tests on the Pavia University dataset. ALBERT is the first classifier, and BERT is the second.

	BERT												Number of Encoders
	without SE				with SE (HyperSLIC)				with SE (HyperSLIC-DBCAN)				
	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9	
ALBERT	26.69/yes	26.41/yes	26.82/yes	26.29/yes	60.47/yes	62.95/yes	60.99/yes	68.96/yes	67.42/yes	71.15/yes	68.09/yes	72.90/yes	**1**
	27.94/yes	26.46/yes	28.01/yes	27.35/yes	61.74/yes	65.01/yes	66.08/yes	67.30/yes	67.54/yes	68.44/yes	69.92/yes	73.88/yes	**3**
	27.22/yes	28.03/yes	27.91/yes	29.56/yes	60.81/yes	62.81/yes	69.05/yes	68.09/yes	67.94/yes	69.22/yes	72.75/yes	74.48/yes	**5**
	28.79/yes	28.93/yes	29.90/yes	29.35/yes	62.73/yes	67.65/yes	68.82/yes	68.73/yes	68.43/yes	72.38/yes	77.27/yes	77.50/yes	**9**
	28.75/yes	28.84/yes	29.54/yes	29.13/yes	63.44/yes	69.55/yes	71.23/yes	71.99/yes	70.45/yes	73.72/yes	77.41/yes	77.54/yes	**12**

TABLE 10. Results of McNemar's tests on the Indian Pines dataset. ALBERT is the first classifier, and BERT is the second.

	BERT												Number of Encoders
	without SE				with SE (HyperSLIC)				with SE (HyperSLIC-DBCAN)				
	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9	
ALBERT	26.34/yes	26.99/yes	27.78/yes	27.34/yes	60.44/yes	60.97/yes	64.72/yes	66.97/yes	67.32/yes	68.76/yes	69.14/yes	73.47/yes	**1**
	26.37/yes	27.18/yes	28.30/yes	28.14/yes	61.64/yes	63.75/yes	65.75/yes	67.28/yes	67.29/yes	69.89/yes	69.65/yes	74.00/yes	**3**
	26.39/yes	27.56/yes	28.63/yes	29.40/yes	63.30/yes	67.05/yes	66.92/yes	70.60/yes	68.87/yes	72.22/yes	73.81/yes	77.43/yes	**5**
	27.03/yes	29.01/yes	29.31/yes	29.77/yes	65.79/yes	67.31/yes	69.27/yes	68.02/yes	69.22/yes	73.27/yes	76.47/yes	77.86/yes	**9**
	27.05/yes	29.30/yes	29.55/yes	29.65/yes	67.80/yes	67.86/yes	71.19/yes	71.59/yes	72.58/yes	73.27/yes	76.89/yes	78.14/yes	**12**

approximately 1.3 times longer than the 3×3 window size. It takes around 1.55 times longer in the BERT model.

The McNemar's test is used to statistically compare the results of the ALBERT and BERT models. McNemar's test is a statistical and objective measure used to compare classification results. This test is commonly used to determine whether or not the results of two classifiers are statistically different. Nonparametric McNemar's test may also be used to compare thematic maps [53] and can be calculated using Equation (9):

$$Z = \frac{Q_{12} - Q_{21}}{\sqrt{Q_{12} + Q_{21}}} \tag{9}$$

McNemar's test employs a different type of 2×2 contingency matrix (Q). The value in the ith row and jth column is represented in Q_{ij}. Q_{12} is the number of samples correctly classified by the first classifier but misclassified by the second one. Q_{21} presents the value obtained by the opposite situation. Their Z value represents the difference between two classifiers. The Chi-square distribution table uses the first degree of freedom and the 5% significant level for comparative purposes. The square root of this number is 1.96. That is, if the computed $|Z|$ value is more than 1.96, the outputs of two classifiers are statistically different. When there is no significant difference between two classifiers (if $|Z| \leq 1.96$), the null hypothesis is defined as H_0:No, while the alternative hypothesis is defined as H_1:Yes, and it is accepted when there is a significant difference between two classifiers (if $|Z| > 1.96$).

Tables 9 and 10 show how statistically our proposed methods, according to McNemar's test results. Null hypotheses are accepted in all circumstances, as shown in the tables. As a result, the ALBERT model outperformed the BERT model both numerically and statistically.

VI. CONCLUSION

In addition to the machine learning methods used in hyperspectral image processing, BERT and ALBERT models, which are increasingly used in natural language processing, are used in this study to classify hyperspectral scenes. To improve hyperspectral image classification performance, a modified HyperSLIC superpixel algorithm for segment embedding information, one of the three inputs given to BERT-based models, is proposed. Then, the HyperSLIC-DBSCAN method is also used by using the DBSCAN algorithm in order to merge the regions with similar spatial contexts found as a result of HyperSLIC segmentation. As a result of experimental studies, it has been seen that BERT-based models, in which spatial information can also be used, can provide better classification accuracies than conventional and deep learning-based 1D/2D convolutional neural network classifiers.

Using segment information in the system helps to overcome the problem of the limited labeled training data. Finding similar pixels without class information provides the classifier with information about all pixels. BERT-based models produce highly accurate results when position and segment information are included. Combining superpixels with the HyperSLIC-DBSCAN method helps reduce noisy superpixels. The classifier learns better as the similarity between pixels increases.

Since the BERT model has more parameters than the ALBERT model, the training time is longer. As a result, the training data needs to be represented with more parameters. Although this depends on the quality of the training data, there may be a limit. Neighborhoods of different window sizes are evaluated in order to overcome the situation where the models have limited information to learn. Provided that the homogeneity is preserved, the classification success increases as the neighborhood area extends.

BERT-based models show high performance in hyperspectral image classification as well as in natural language processing. Studies in natural language processing show that models with different architectures such as ELECTRA, ERNIE, and GPT are also successful. BERT-based models learn by evaluating before and after the word in the sentence.

Because of this feature, when analyzing a pixel, it is easier to evaluate all its neighbors. In other architectures, different approaches need to be constructed to classify hyperspectral images.

REFERENCES

[1] J. A. Gualtieri and R. F. Cromp, "Support vector machines for hyperspectral remote sensing classification," in *Proc. 27th AIPR Workshop, Adv. Comput.-Assist. Recognit.*, 1999, pp. 221–232.

[2] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[3] S. Kawaguchi and R. Nishii, "Hyperspectral image classification by bootstrap AdaBoost with random decision stumps," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 11, pp. 3845–3851, Nov. 2007.

[4] G. Camps-Valls and L. Bruzzone, "Kernel-based methods for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 6, pp. 1351–1362, Jun. 2005.

[5] M. Fauvel, "Spectral and spatial methods for the classification of urban remote sensing data," Ph.D. dissertation, Institut National Polytechnique de Grenoble-INPG, Université d'Islande, Grenoble, France, 2007.

[6] G. Camps-Valls, L. Gomez-Chova, J. Muñoz-Marí, J. Vila-Francés, and J. Calpe-Maravilla, "Composite kernels for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 3, no. 1, pp. 93–97, Jan. 2006.

[7] J. A. Benediktsson, J. A. Palmason, and J. R. Sveinsson, "Classification of hyperspectral data from urban areas based on extended morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 480–491, Mar. 2005.

[8] X. Huang and L. Zhang, "A comparative study of spatial approaches for urban mapping using hyperspectral ROSIS images over Pavia City, northern Italy," *Int. J. Remote Sens.*, vol. 30, no. 12, pp. 3205–3221, 2009.

[9] Y. Tarabalka, J. Chanussot, J. A. Benediktsson, J. Angulo, and M. Fauvel, "Segmentation and classification of hyperspectral data using watershed," in *Proc. IGARSS - IEEE Int. Geosci. Remote Sens. Symp.*, 2008.

[10] A. N. Evans and X. U. Liu, "A morphological gradient approach to color edge detection," *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1454–1463, Jun. 2006.

[11] Y. Tarabalka, J. Chanussot, and J. A. Benediktsson, "Classification based marker selection for watershed transform of hyperspectral images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, vol. 3, Jul. 2009, pp. III–105.

[12] Y. Tarabalka, J. Chanussot, and J. A. Benediktsson, "Classification of hyperspectral images using automatic marker selection and minimum spanning forest," in *Proc. 1st Workshop Hyperspectral Image Signal Process., Evol. Remote Sens.*, Aug. 2009, pp. 1–4.

[13] J. Stawiaski, "Mathematical morphology and graphs: Application to interactive medical image segmentation," Ph.D. dissertation, Ecole Nationale Superieure des Mines de Paris, Paris, France, 2008.

[14] N. L. Kazanskiy, P. G. Serafimovich, and E. A. Zimichev, "Spectral-spatial classification of hyperspectral images with k-means++ partitional clustering," in *Optical Technologies for Telecommunications*. Bellingham, WA, USA: International Society for Optics and Photonics, 2015, Art. no. 95330M.

[15] B. Zhang, S. Li, C. Wu, L. Gao, W. Zhang, and M. Peng, "A neighbourhood-constrained k-means approach to classify very high spatial resolution hyperspectral imagery," *Remote Sens. Lett.*, vol. 4, no. 2, pp. 161–170, Feb. 2013.

[16] L. Fang, S. Li, W. Duan, J. Ren, and J. A. Benediktsson, "Classification of hyperspectral images by exploiting spectral–spatial information of superpixel via multiple kernels," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 12, pp. 6663–6674, Dec. 2015.

[17] L. Fang, S. Li, X. Kang, and J. A. Benediktsson, "Spectral–spatial classification of hyperspectral images with a superpixel-based discriminative sparse model," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 8, pp. 4186–4201, Aug. 2015.

[18] J. Bai, W. Shi, Z. Xiao, A. C. Regan, T. A. A. Ali, Y. Zhu, R. Zhang, and L. Jiao, "Hyperspectral image classification based on superpixel feature subdivision and adaptive graph structure," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–15, 2022.

[19] C. Zhao, B. Qin, S. Feng, and W. Zhu, "Multiple superpixel graphs learning based on adaptive multiscale segmentation for hyperspectral image classification," *Remote Sens.*, vol. 14, no. 3, p. 681, Jan. 2022.

[20] F. Xie, Q. Gao, C. Jin, and F. Zhao, "Hyperspectral image classification based on superpixel pooling convolutional neural network with transfer learning," *Remote Sens.*, vol. 13, no. 5, p. 930, Mar. 2021.

[21] C. Chen, W. Li, H. Su, and K. Liu, "Spectral–spatial classification of hyperspectral image based on kernel extreme learning machine," *Remote Sens.*, vol. 6, no. 6, pp. 5795–5814, 2014.

[22] W. Li, C. Chen, H. Su, and Q. Du, "Local binary patterns and extreme learning machine for hyperspectral imagery classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 7, pp. 3681–3693, Jul. 2015.

[23] M. Ahmad, S. Shabbir, S. K. Roy, D. Hong, X. Wu, J. Yao, A. M. Khan, M. Mazzara, S. Distefano, and J. Chanussot, "Hyperspectral image classification-traditional to deep models: A survey for future prospects," 2021, *arXiv:2101.06116*.

[24] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016.

[25] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *J. Sensors*, vol. 2015, pp. 1–12, Jan. 2015.

[26] J. M. Haut, M. E. Paoletti, J. Plaza, J. Li, and A. Plaza, "Active learning with convolutional neural networks for hyperspectral image classification using a new Bayesian approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6440–6461, Nov. 2018.

[27] X. Yang, Y. Ye, X. Li, R. Y. K. Lau, X. Zhang, and X. Huang, "Hyperspectral image classification with deep learning models," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5408–5423, Sep. 2018.

[28] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1349–1362, Mar. 2015.

[29] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.

[30] J. Lin, R. Ward, and Z. J. Wang, "Deep transfer learning for hyperspectral image classification," in *Proc. IEEE 20th Int. Workshop Multimedia Signal Process. (MMSP)*, Aug. 2018, pp. 1–5.

[31] L. Zhu, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Generative adversarial networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5046–5063, Sep. 2018.

[32] Y. Zhan, D. Hu, Y. Wang, and X. Yu, "Semisupervised hyperspectral image classification based on generative adversarial networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 2, pp. 212–216, Feb. 2017.

[33] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3639–3655, Apr. 2017.

[34] H. Wu and S. Prasad, "Convolutional recurrent neural networks for hyperspectral data classification," *Remote Sens.*, vol. 9, no. 3, p. 298, Mar. 2017.

[35] R. Hang, Q. Liu, D. Hong, and P. Ghamisi, "Cascaded recurrent neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5384–5394, Aug. 2019.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[37] J. He, L. Zhao, H. Yang, M. Zhang, and W. Li, "HSI-BERT: Hyperspectral image classification using the bidirectional encoder representation from transformers," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 1, pp. 165–178, Jan. 2019.

[38] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," 2019, *arXiv:1909.11942*.

[39] D. Hong, Z. Han, J. Yao, L. Gao, B. Zhang, A. Plaza, and J. Chanussot, "SpectralFormer: Rethinking hyperspectral image classification with transformers," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–15, 2021.

[40] X. He, Y. Chen, and Z. Lin, "Spatial-spectral transformer for hyperspectral image classification," *Remote Sens.*, vol. 13, no. 3, p. 498, Jan. 2021.

[41] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[42] P. Kovesi. (2013). *Image Segmentation Using SLIC Superpixels and DBSCAN Clustering*. Accessed: Dec. 17, 2021. [Online]. Available: https://bit.ly/3LUPaWS

[43] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discov. Data Mining*, 1996, pp. 226–231.

[44] Z. Wang and A. C. Bovik, "A universal image quality index," *IEEE Signal Process. Lett.*, vol. 9, no. 3, pp. 81–84, Aug. 2002.

[45] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," 2016, *arXiv:1606.05250*.

[46] J. Lei Ba, J. Ryan Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[48] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

[49] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12 no. 10, pp. 2825–2830, 2011.

[51] J. A. Benediktsson and P. Ghamisi, *Spectral-Spatial Classification of Hyperspectral Remote Sensing Images*. Norwood, MA, USA: Artech House, 2015.

[52] B. Rasti, D. Hong, R. Hang, P. Ghamisi, X. Kang, J. Chanussot, and J. A. Benediktsson, "Feature extraction for hyperspectral imagery: The evolution from shallow to deep: Overview and toolbox," *IEEE Geosci. Remote Sens. Mag.*, vol. 8, no. 4, pp. 60–88, Dec. 2020.

[53] G. M. Foody, "Thematic map comparison," *Photogramm. Eng. Remote Sens.*, vol. 70, no. 5, pp. 627–633, 2004.

IBRAHIM ONUR SIGIRCI received the Ph.D. degree in computer engineering from Yildiz Technical University (YTU), Istanbul, Turkey. He is currently a member of the Signal and Image Processing Laboratory (SIMPLAB, www.simplab.yildiz.edu.tr), Department of Computer Engineering, YTU. His main research interests include spectral-spatial hyperspectral image classification, biomedical image processing, computer vision, machine learning, and algorithms.

GOKHAN BILGIN received the B.Sc., M.Sc., and Ph.D. degrees in electronics and telecommunication engineering from Yildiz Technical University (YTU), Istanbul, Turkey, in 1999, 2003, and 2009, respectively. He has worked as a Postdoctoral Researcher at the Computer Aided Diagnosis and Knowledge Discovery Laboratory (CADKD), Department of Computer and Information Science, Indiana University–Purdue University Indianapolis (IUPUI). He is currently working as an Associate Professor with the Department of Computer Engineering, YTU. He is the Founder of the Signal and Image Processing Laboratory (SIMPLAB, www.simplab.yildiz.edu.tr), Department of Computer Engineering, YTU. His research interests include image and signal processing, machine learning, and pattern recognition with applications to biomedical engineering and remote sensing.