A Branching Spatio-Spectral Dimensional Reduction Model for Hyperspectral Image Classification and Change Detection

Menilk Sahlu Bayeh[1(✉)], Anteneh Tilaye Bogale[2], Yunkoo Chung[2], Kirubel Abebe Senbeto[3], and Fetlewerk Kedir Abdu[2]

[1] Saint Mary's University, Addis Ababa, Ethiopia
[2] Adama Science and Technology University, Adama, Ethiopia
[3] Woldia University, Woldia, Ethiopia

Abstract. In this paper, a branching convolutional encoder (BCE)-based spatio-spectral hyperspectral image dimensionality reduction model is presented. The architecture consists of a pointwise separable convolution to extract spectral features, and a two-dimensional convolution network to filter spatial features. Later, these two features are fused and fed into a decoder network which attempts to reconstruct the original image. This network is trained in a similar fashion to autoencoders, using a loss function to track the similarity between the original and the reconstructed image. Classification and change detection are important applications of hyperspectral images. The branching convolutional encoder is used together with classification and change detection models to demonstrate its feature representation performance – since the raw image has redundant features and poor interclass separability. The performance of the proposed dimensionality reduction model is compared with a spatial convolutional encoder and a densely-connected encoder. Classification accuracy reaches over 90% on all the datasets which out-performs the comparative methods. Moreover, the branching encoder's representation power is observed with the change detection model as the rate of accuracy reaches over 99% for the Hermiston City-data. This research demonstrably presents the success of a branching convolutional dimensionality encoder for classification and change detection applications.

Keywords: Dimensionality reduction · Autoencoder · Hyperspectral images

1 Introduction

Multiclass classification and change detection are important applications of hyperspectral images used to track changes in a specified geographic location [1, 2]. Multitemporal hyperspectral images are used to track changes in precision agriculture, mineral mining and exploration, security surveillance for military applications, monitoring environment for hazard and natural disasters, etc. [3–5]. Considering the hundreds of bands and redundant information available in HSIs, it is quite evident that these images are useful for spectral analysis to track changes in a desired area in more

M. L. Berihun (Ed.): ICAST 2021, LNICST 411, pp. 532–549, 2022.
https://doi.org/10.1007/978-3-030-93709-6_36

detail. However, this advantage also introduces computational and algorithmic complexity. Performing computation on high dimensional data is resource intensive and takes a long time. This complexity introduced by high dimensional data is called "the curse of dimensionality" [6–9].

This problem has been researched for several decades years now with different techniques ranging from complex feature extractors and change detection algorithms, going through machine learning algorithms in the 2000s and most recently with deep learning architectures. Feature extraction and selection algorithms were employed to reduce the dimension of the original hyperspectral data making the output easy to manipulate for further applications [10]. However, most feature extraction algorithms do not consider both the spatial and spectral information available in HSIs when forming a feature representation.

In this research efficient ways of autoencoder based dimensionality reduction (DR) techniques were explored that involve a deep spatio-spectral feature extractor that creates an efficient latent feature representation for classification and change detection applications. The efficiency of the feature extractor is tested via different classification algorithms. After this has been demonstrated the feature extractor is used with the change detection algorithm. These two are trained in a coupled manner and the effectiveness of the entire framework is tested by different performance metrics. Finally, the framework's performance is compared with existing algorithms to prove its effectiveness of existing techniques.

The rest of the paper is organized as follows. Section 2 introduces previous dimensionality reductions approaches used and the most recent deep learning-based feature extraction techniques for the classification and change detection application. Section 3 presents the detailed of the branching convolutional encoder model architecture along with the classifier and change detection models used to test the DR module's effectiveness. Section 4 includes the details of data preparation and training. It also consists of training results and comparison with other comparative methods. Section 5 concludes the paper and makes recommendations for future works.

2 Dimensionality Reduction for Hyperspectral Images

Dimensionality reduction is the change of information from a high-dimensional space into a low-dimensional space with the goal that the low-dimensional portrayal holds some important properties of the first content, preferably near its natural measurement. Popular dimensionality reduction techniques are commonly divided into linear and nonlinear approaches. Approaches can also be divided into feature selection and feature extraction based on how features are treated. Dimensionality reduction can be utilized for noise reduction, information perception, bunch investigation, or as a transitional step to facilitate different tasks [6, 7].

Lloyd Windrim et al. propose a stacked autoencoder based unsupervised feature learning for Dimensionality reduction [11]. The feature learning algorithm projects feature representation in to a latent space so as to reduce the dimension of the original data to be used for classification application. Moreover, they integrated an information theoretic measure based spectral information divergence metric to make the reconstructed output spectrally similar with the original input. Also, they used Cosine Angle Similarity (CSA) measure to train the autoencoder's latent representation efficient. However, their method does not consider the spatial aspect of the representation as they used a one-dimensional (1D) stacked autoencoder which learns to reconstruct the original spectra.

Ayma et al. propose an orthogonal autoencoder dimensionality reduction approach for classification of hyperspectral images [12]. They introduce an orthogonal reconstruction error which is defined as the sum of mean squared error between the original and reconstructed output, and the mean squared error between the latent variable product and an identity matrix *I*. The orthogonality between components in the latent space is ensured by the loss function and the training optimizer so that the orthogonality of latent components improves classification performance. They tested their models on the Pavia University, Kennedy Space Center, and Botwana hyperspectral images. However, one obvious draw back in this method is that it does not consider spatial and spatial components during the feature representation.

Filtering approach by using 1D pooling has been proposed by Paul and Chaki to reduce/select important spectral features while reducing the dimension of the original image [13]. The one benefit of this method is that it clearly reduces the computation time taken while reducing the dimension and is less complex in its computation. However, this technique is also in line with the 1D techniques that focus on spectral information primarily, and require reshaping the original input to two-dimensional (2D) pixel vector array which jumbles spatial information along the process.

The alternative candidates to 1D autoencoders are convolutional autoencoder that capture spatial information from the hundreds of channels present in the image [14]. Mei et al. propose a three-dimensional (3D) convolutional autoencoder spatial-spectral feature learning for hyperspectral image classification application [15]. Elementwise 3D convolutions, 3D pooling and batch normalization have been included in this work. The encoder is trained together with a decoder counter-part that attempts to reconstruct the original input from the spatial-spectral feature representation. They tested the model's performance on a support vector machine (SVM) classifier to test the feature representation performance allowing the components to be easily classified.

There is a clear progression from 1D multilayer perceptron autoencoder techniques to higher dimensional convolutional autoencoder to train encoder to create latent representation learning either from hundreds of bands spatially, or from both the spatial and spectral representation. However, separate treatment of the spatial and spectral components is yet to be seen as spectral relationships, spatial relationships and spatio-spectral relationships need to successively considered when designing feature extractor-based dimensionality reduction techniques. BCE aims to overcome this shortage by incorporating a spatio-spectral dimensionality reduction framework.

3 Branching Convolutional Autoencoder

The numerous literatures surveyed in the previous chapter show that most techniques do not effectively combine spatial-spectral representation for dimensionality reduction. This requires a combined dimensionality reduction architecture that integrates both the spatial-spectral features. Branching Convolutional Encoder aims to alleviate this problem by designing to separate convolutional encoder to represent the spatial and spectral information from the original hyperspectral cube. The spectral encoder consists of a point wise convolution that performs 1D convolutions with the pixel vectors which correspond to the values present at a single pixel slice having hundreds of spectral bands.

Since the original cube is in reflectance value corresponding to each and every spectrum, the pixel values need to undergo normalization to avoid gradient explosion during training. After normalization, patches of the original cube will be taken to make the dataset required to train the network. These patches will be later fed in to the encoder section of the autoencoder. The encoder section of branching autoencoder performs separate convolutions on the patches through a spectral and a spatial encoders sub-section.

Later, the output of these encoders is fused together to form the spatial-spectral representation. The output of these feature space is fed into a convolutional decoder that attempts to reconstruct the original input. This output will be compared on with the original patch to compare whether the reconstructed output is similar to the original or not. Training of the autoencoder took place until the reconstruction closely resembled the original. Finally, the encoder section was taken out and used as a feature extractor for classification and change detection application.

The 1D convolution operation convolves a kernel k having dimensions ($1 \times B$), where B is the number of channels/bands. This kernel is convolved with each and every pixel vector (slice) of the hyperspectral cube give a 2D output of shape ($M \times N$) where M and N the number of row and column pixel count. 2D-convolution (Conv2D) layer of the Keras library initializes numerous kernels of different initializations where after each 1D kernel has been convolved with the input the output shape becomes (M, N, K). K adjusts the output dimension of the convolution without affecting M and N.

Moving on to the spatial encoder section, 2D convolution is performed to capture spatial information and reduce the size of the output whose depth is determined by the number of kernels present. One convolution operation with a single kernel amongst the total count K' gives output shape specified by the number of strides, zero padding and kernel size. Differing kernel size are used to effectively capture the spatial features present on different neighborhood size. Stacking up these convolutions also stacks up the feature representations where at a certain point the output of the last layers contains some information about each and every pixel vector before it. This enhances spatial feature representation on an integrated 2D level (Fig. 1).

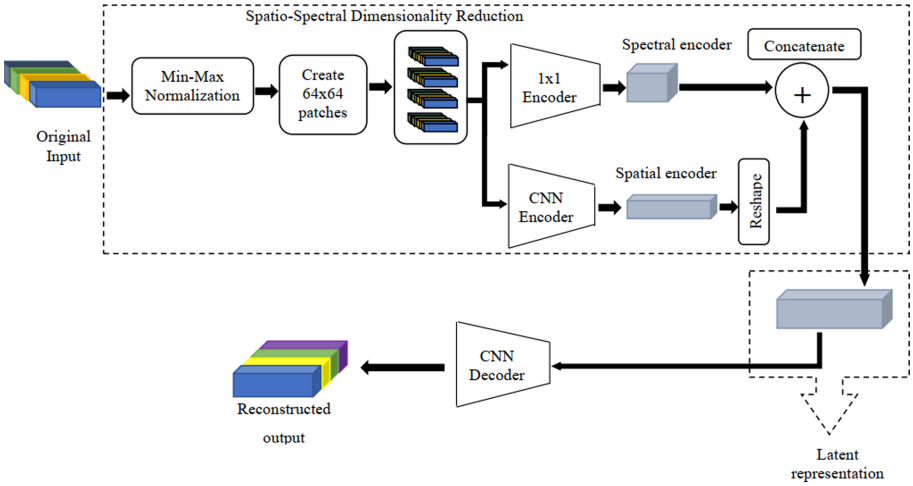

Fig. 1. Branching convolutional autoencoder architecture

The third step is fusing the spatial and spectral representation via a concatenation step. The concatenated output contains both the spectral and spatial features without any significant omission. This concatenated spatio-spectral representation goes to the decoder which consists of deconvolutions (transposed convolution) and up sampling steps that attempt to reconstruct the original dimensions input the encoder network. The spatial dimensions are controlled by the kernel size, stride, and zero paddings, while the depth is maintained by the number of pixels in each ConvTranspose2D layer. Using 1D and 2D convolution will reduce the number of operations done in 3D densely connected models. Training of the encoder-decoder architecture is controlled by a loss function that computes error between the original and the reconstructed output.

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1N} \\ X_{21} & X_{22} & X_{23} & \dots & X_{2N} \\ X_{31} & X_{32} & X_{33} & \dots & X_{3N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X_{M1} & X_{M2} & X_{M3} & \cdots & X_{MN} \end{bmatrix}_{M \times N} \tag{1}$$

Unfolding of the matrix above reveals the vectorized representation and the pixel elements on the B bands available in the image. Equation (1) gives the flattened representation of the input where each and every element in the above matrix is expanded to reveal the corresponding pixel elements.

$$\left\{\begin{array}{l} X_{11} = \left[\begin{array}{ccccc} X_{11}^1 & X_{11}^2 & X_{11}^3 & \cdots & X_{11}^B \end{array}\right]_{I\times B} \\ X_{12} = \left[\begin{array}{ccccc} X_{I2}^1 & X_{I2}^2 & X_{12}^3 & \cdots & X_{I2}^B \end{array}\right]_{I\times B} \\ \quad \vdots \\ X_{MN} = \left[\begin{array}{ccccc} X_{MN}^1 & X_{MN}^2 & X_{MN}^3 & \cdots & X_{MN}^B \end{array}\right]_{1\times B} \end{array}\right\} \tag{2}$$

Pointwise convolution between the elements and kernel gives the out where each point in the matrix is the convolution between the kernel and the pixel vectors at the corresponding points. The kernel width has to be set equal to the number of the bands as in Eq. (3) for the first step so that the output of the convolution becomes a single matrix as shown in Eq. 4.

$$K_1 = \left[\begin{array}{ccccc} K_1^1 & K_1^2 & K_1^3 & \cdots & K_1^B \end{array}\right]_{(I\times B)} \tag{3}$$

Equation (4) demonstrates the convolution between the kernel vectors and the pixel vectors.

$$K_1 * X = \left[\begin{array}{c} k_1 * X_{11}^T \\ k_1 * X_{12}^T \\ \vdots \\ k_1 * X_{MN}^T \end{array}\right] \tag{4}$$

Where, the convolution between the several kernels and the input becomes:

$$K_1 * X_{11}^T = \left[k_1^1 \times X_{11}^T + k_1^2 \times X_{11}^2 + \cdots + k_1^B \times X_{11}^B\right] \times \frac{1}{B} \tag{5}$$

The final depth of the convolution step is determined by the number of kernels specified which successively drops in equal number of steps. Therefore, the output has the same shape as the input show in Eq. (6) but different depth. Later the convolution outputs will be multiplied by weights and added with the biases specified.

$$C_1 = \left[\begin{array}{ccccc} k_1 * X_{11} & k_1 * X_{12} & k_1 * X_{13} & \ldots & k_1 * X_{1N} \\ k_1 * X_{21} & k_1 * X_{22} & k_1 * X_{23} & \ldots & k_1 * X_{2N} \\ k_1 * X_{31} & k_1 * X_{32} & k_1 * X_{33} & \ldots & k_1 * X_{3N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ k_1 * X_{M1} & k_1 * X_{M2} & k_1 * X_{M3} & \cdots & k_1 * X_{MN} \end{array}\right] \tag{6}$$

3.1 Densely-Connected Classifier

The conventional setup of classifiers used in image classification networks has been used to design the classifier scheme. Once the dimensionality reduction model has been

trained on the available data it will be integrated with the densely connected layers that consist of SoftMax layer that outputs class probabilities on the given it input classifying it to the number of classes available for that dataset. Cross categorical entropy has been used to train this model and update the weights of the network. This model will be used to test performance of BCE and the comparative dimensionality reduction techniques (Densely connected network and a 2D convolutional autoencoder). The diagrammatic representation of the architecture of this network is given below.

3.2 Adapted Convolutional LSTM-Based Change Detection Architecture

The convolutional Long Short-term Memory (LSTM) architecture proposed by Ahram Song et al. is used for determining changes between two multitemporal hyperspectral images [16]. This architecture is trained using paired patches from two multitemporal images and the resulting output is fed in to a densely connected multiclass classification network to plot the multiclass change detection map that not only gives the changed areas but also the classes to which each changed pixel belongs to. A conventional Convolutional LSTM (ConvLSTM) block is used here extract temporal information between the bitemporal images. The cascaded ConvLSTM layers extract information from a time distributed that simultaneously applies a 2D convolutional operation and extracts features which might give information about changes in the bitemporal images. Several components make up the ConvLSTM block such as sigmoid activation represented by σ, tanh activation block, h-blocks that represent hidden layer input from previous slices, forget gates represented by f, pointwise addition and multiplication denoted by circled plus and asterisk sign, etc. (Fig. 2).

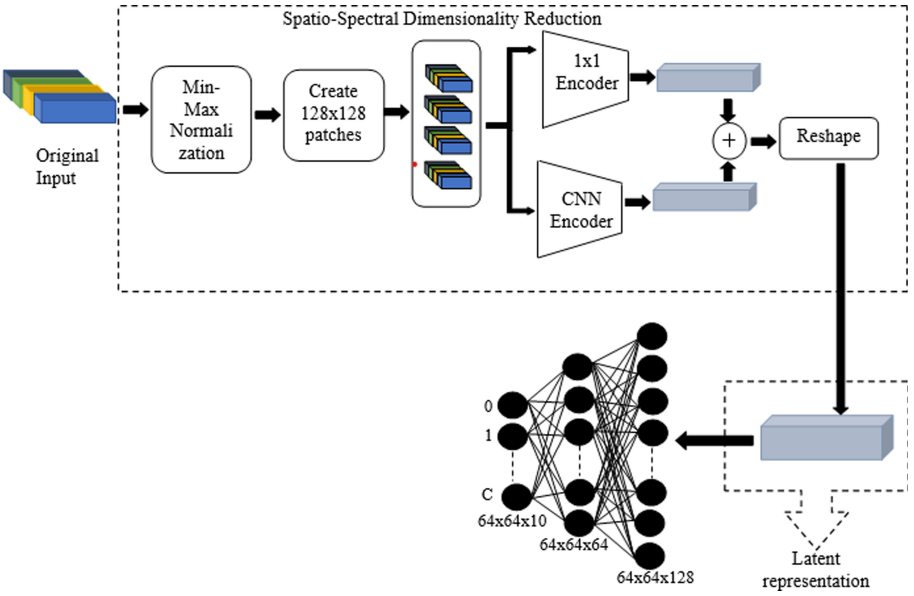

Fig. 2. Branching Convolutional Encoder concatenated with a densely connected classifier (e.g., for the Pavia University dataset)

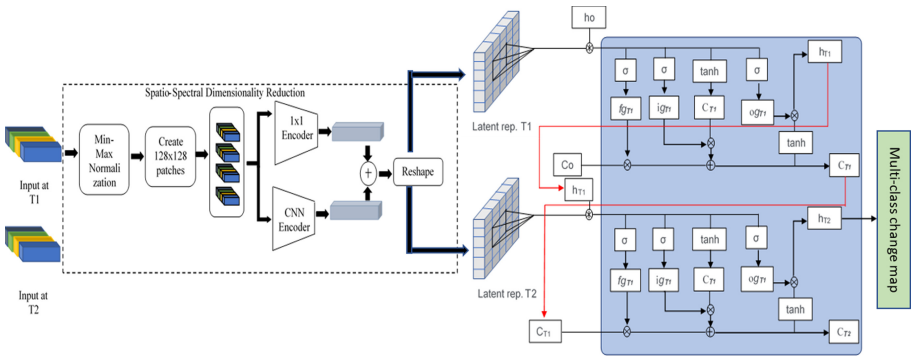

Fig. 3. BCE concatenated with a change detection model

The architecture given on Fig. 3 is integrated with the dimensionality reduction module given above and trained in an end-to-end manner to generate the multiclass change detection map for each patch. Two images from the same spatial slice will be fed in to the dimensionality reduction model and fed to the change detection model, so that change detection model learns from the feature representations of the change detection model. Finally, the densely connected layer at the end of the ConvLSTM model outputs a feature classification map that contains pixels where changes took place and what kind of images took place in those pixels.

4 Implementation Details and Results

In this section the datasets used, the details of implementation step, and the results of the training are provided.

4.1 Branching Convolutional Autoencoder Implementation Details

First the Branching Convolutional Autoencoder (BCAE) is implemented with branching encoders whose outputs are reshaped to match each other and fused together to form the latent representation. Figure 4 below shows the implementation level view of the BCAE model. The decoder consists of an ascending combination of filter size as it is the convention to use varying kernel size in most papers that use convolutional neural network architecture. Moreover, the exact detail showing how the two encoders are implemented in TensorFlow 2.x is described on Table 1 and Table 2 respectively. Table 1 shows that the spectral encoder is composed of convolution layers performing point-wise convolutions on the input. While the spatial encoder has the regular 2D convolution of descending kernel sizes.

4.2 Datasets

Four popular datasets were used first for training the autoencoder for each corresponding dataset, and then used later for the classification task since they have a multiclass

annotation available. Pavia Center Scene dataset[1] has 102 spectral bands, and each raster is composed of 1096 * 715 pixels. Pavia Center Scene dataset has nine classes with one extra class consisting of unclassified pixels. This image is used for demonstrating the representation power of the hyperspectral images and later for the classification task. The Pavia University dataset is closely related to the Pavia Center dataset as they were both collected during the same flight. Compared with the Pavia center it has a smaller spatial dimension (610 × 340) but has one extra channel than the Pavia Center dataset. Both of them have the same number of classes and class descriptions.

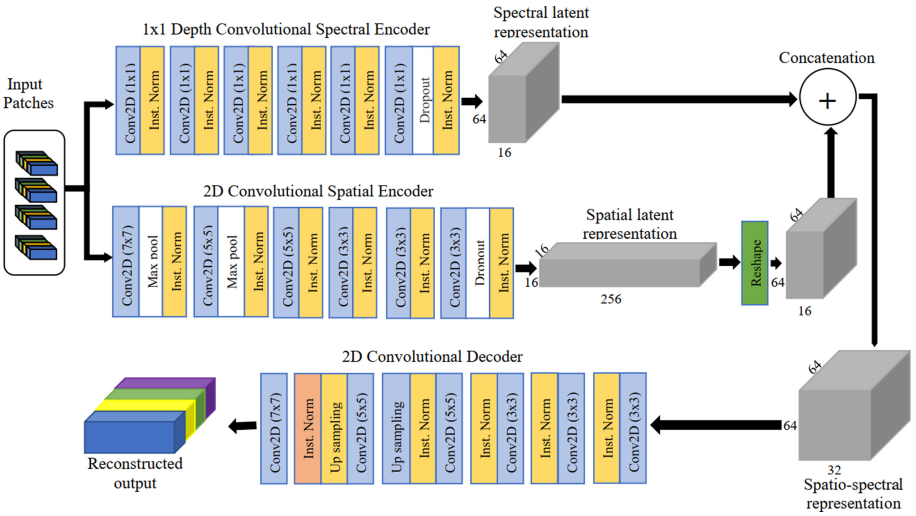

Fig. 4. Implementation-level view of BCAE model (for the PaviaC dataset)

The Salinas Valley scene consists of 16 classes plus one extra unclassified class labelled as zero. The surface reflectance values are available over a grid of 517 × 217 pixels. This dataset is used for testing the feature representation power of the dimensionality reduction models and for testing classification performance. This Kennedy Space Center (KSC) image was captured in the year 1996. The KSC scene image band has been reduced to 176 bands after the removal of water absorption bands. The hyperspectral cube's image is dark and hard to see due to the nature of the conditions when the image was taken – this makes the reconstruction task challenging as the contrast difference between the pixels is not sparse enough to differentiate. The Hermiston city bitemporal images has 242 spectral channels each of which have a 390 × 200 dimension monochromatic images. The represents crop transition of different classes in Hermiston city where center pivot irrigation creates the circular patterns in the image due to water supply.

[1] PaviaU, PaviaC, KSC, and Salinas Scene datasets will be found at http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes.

Table 1. TensorFlow implementation level summary for spectral encoder

Convolution operation	Filter size	No. of filters	Padding size	Stride size	Input size	Output size
Conv2D	(1 × 1 × 102)	85	Zero Pad	(1, 1)	64 × 64 × 102	64 × 64 × 85
Conv2D	(1 × 1 × 85)	68	Zero Pad	(1, 1)	64 × 64 × 85	64 × 64 × 68
Conv2D	(1 × 1 × 68)	51	Zero Pad	(1, 1)	64 × 64 × 68	64 × 64 × 51
Conv2D	(1 × 1 × 51)	34	Zero Pad	(1, 1)	64 × 64 × 51	64 × 64 × 34
Conv2D	(1 × 1 × 34)	16	Zero Pad	(1, 1)	64 × 64 × 34	64 × 64 × 16

Table 2. TensorFlow implementation level summary of the spatial encoder

Operation	Filter size	No. of filters	Padding size	Stride size	Input size	Output size
Conv2D	(7 × 7 × 102)	16	Same	(1, 1)	64 × 64 × 102	64 × 64 × 16
MaxPool2D	(2 × 2)	–	Same	(1, 1)	64 × 64 × 16	32 × 32 × 16
Conv2D	(5 × 5 × 16)	64	Same	(1, 1)	32 × 32 × 16	32 × 32 × 64
MaxPool2D	(2 × 2)	–	Same	(1, 1)	32 × 32 × 64	16 × 16 × 64
Conv2D	(5 × 5 × 64)	112	Same	(1, 1)	16 × 16 × 64	16 × 16 × 112
Conv2D	(3 × 3 × 112)	160	Same	(1, 1)	16 × 16 × 112	16 × 16 × 160
Conv2D	(3 × 3 × 160)	208	Same	(1, 1)	16 × 16 × 160	16 × 16 × 208
Conv2D	(3 × 3 × 208)	256	Same	(1, 1)	16 × 16 × 208	16 × 16 × 256

4.3 Comparative Models

The encoder sections of the following two comparative autoencoder models were taken as a feature extractor and used for classification and change detection. Their performance was recorded and included below for comparison purpose.

4.3.1 Densely Connected Autoencoder

A densely connected autoencoder was designed and implemented as an autoencoder. The input to the autoencoder is not flattened as with stacked autoencoders, but is the raw patched hyperspectral image (e.g., 64 × 64 × 102 for the PaviaC Image). Later the autoencoder outputs the reconstructed output at the end. In a manner similar to BCAE the dimension at the feature space is consistent so that there is not any discrepancy (e.g., 64 × 64 × 32 for the PaviaC image).

4.3.2 Convolutional Autoencoder

This is a plain convolutional autoencoder where the encoder section consists of max pooling to reduce the size of the image, while the output has a series of interleaved upsampling layers combined with convolution layers. The dimension of the data at the feature space is the same with the BCAE.

4.4 Implementation Details

In order to use the encoder section of the three autoencoder models including the BCAE the autoencoders first had to be trained until the loss between the original and the reconstructed patches became very small. For the four images mentioned above the original images were normalized and patched to form a dataset that can be used to train the networks. Out of the four datasets 85% of the data was used to train the models and 15% to test the model performance. Once the autoencoders were trained their encoders were sectioned out and integrated with the classification and change detection models. The classification and change detection models were trained after being integrated with the dimensionality reduction model (feature extractor). Optimal hyperparameters for the BCAE and the other networks was determined using grid search. All the models were trained on a computer having NVIDIA RTX 2070 Ti installed.

4.5 Evaluation Metrics

There are three clearly distinct tasks in this paper namely training the autoencoder models for via reconstruction, training the classifier, and training the change detection model. For the first task similarity metrics L1 and L2 losses were used to measure whether the reconstructed image is similar to the original or not. Also, spectral similarity measures Spectral Angle (SA) and Cosine of Spectral Angle (CSA) were used to measure whether the reconstructed images are spectral consistent with their original counter parts. Since the classification and the change detection models have a multi-class map as their output Accuracy, Recall, Precision, and F1-Score are used to measure their performance.

4.6 Experimental Results

Here the results of the three experiments mentioned above are provided with the necessary visualizations.

4.6.1 Autoencoders Reconstruction Results

Each of the autoencoder models BCAE, Densely Connected Autoencoder (DCAE), and Convolutional Autoencoder (CAE) were trained on the four datasets. Finally, they were tested for their reconstruction ability on the four datasets using the similarity evaluation metrics. Out the three models the BCAE has the smallest number of additional artifacts on the reconstructed image, visually resembles the original input. Some significant results to note are the densely connected autoencoder (DCAE) performs the worst on the KSC dataset because of the narrow contrast in the image. Since the DCAE simply performs linear activation and does not have any significant feature transformation, its performance is the lowest. As for the CAE it has a good performance compared to the DCAE, but there are noticeable additional artifacts (grid like and some blurring lines) on the reconstructed image especially on the KSC dataset. The BCAE also has noticeable artifacts on the KSC dataset, but has a better visual similarity to the original than the other two (Fig. 5).

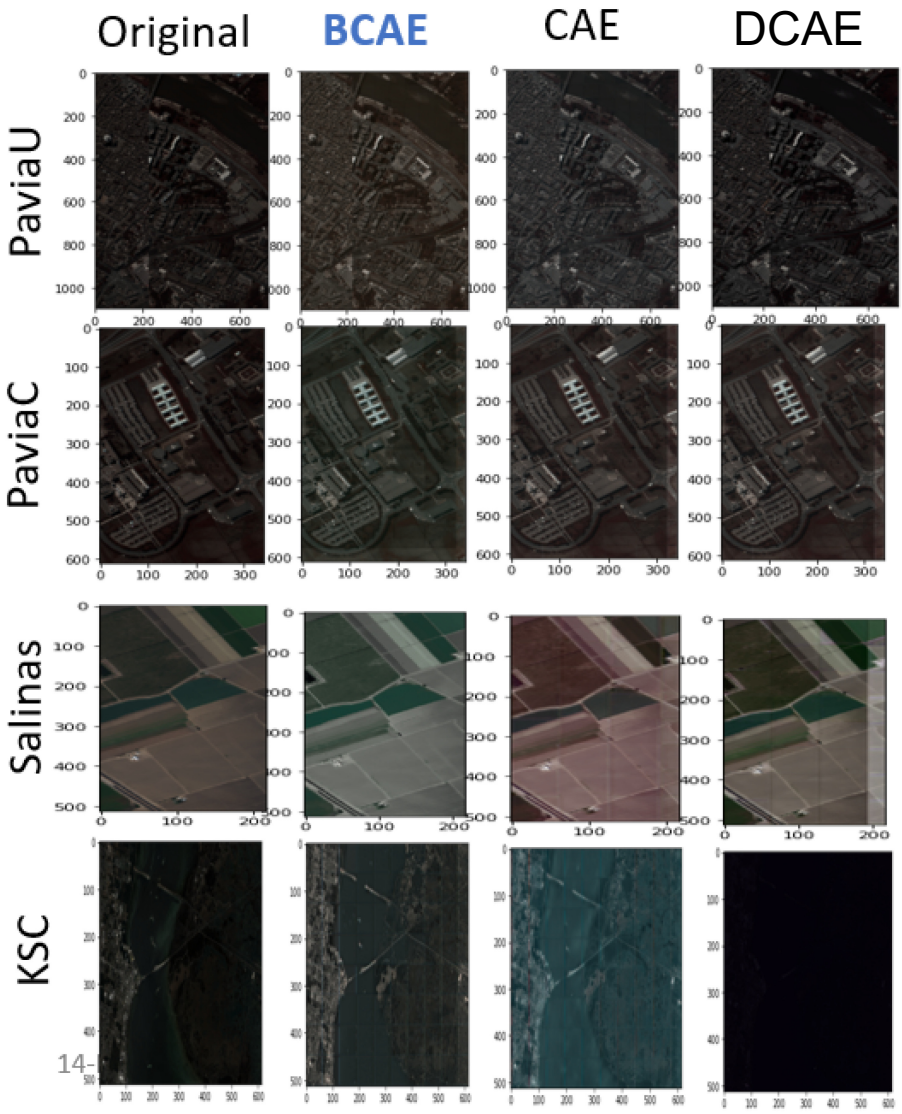

Fig. 5. Reconstruction results for the three autoencoders using the datasets

Taking a closer look at the reconstruction task via the performance evaluation metrics the BCAE reconstruction outputs have the overall small spatial (L1 and L2) and spectral (SA and CSA) losses. However, there are cases where the BCAE losses are not the smallest. This, nonetheless is better understood further by the classification and

change detection metrics. The smallest results for each dataset are given in bold. With the exception of two instances BCAE outperforms the others. Table 1 summarizes the reconstruction results (Table 3).

Table 3. Reconstruction performance evaluation for the three autoencoders

Dataset	DR model	Evaluation metrics			
		L1-metric	L2-metric	SA	CSA
KSC	CAE	**0.00223**	0.04434	1.56569	0.9949
	DCAE	0.00274	**0.04333**	1.56571	0.9949
	BCAE	0.00224	**0.04433**	**1.56562**	**0.9948**
PaviaC	CAE	0.01549	0.02418	1.56113	0.9904
	DCAE	0.01191	0.01817	1.56111	0.9904
	BCAE	**0.01083**	**0.01667**	**1.56107**	**0.9903**
PaviaU	CAE	0.01687	0.03106	1.56112	0.9903
	DCAE	0.02116	0.03250	1.56113	**0.9902**
	BCAE	**0.01083**	**0.01667**	**1.56107**	0.9903
Salinas	CAE	0.01010	0.02351	1.56591	**0.9951**
	DCAE	0.01245	0.03090	**1.56590**	**0.9951**
	BCAE	**0.00549**	**0.01008**	1.56591	**0.9951**

4.6.2 Classification Results

Once the autoencoders finished their training, their encoders (Branching Convolutional Encoder (BCE), Densely Connected Autoencoder (DCE), and Convolutional Encoder (CE)) were sectioned out and used together with the classifier as a feature extractor. Figure 6 below shows the classification map output for three datasets along with the original ground truth map comparison. It can be seen from the image that the number of artifacts present in BCE are smaller compared with DCE and CE. DCE results in the lowest performance out of the three as it's feature representation performance is the lowest. Tabular summary given on Table 4 shows the overall performance of the classifier network with respect to accuracy, recall, precision, and F1-score. With one exception for the Salinas dataset where the CAE has the highest recall, BCAE outperforms the rest with significant margin – which is a clear testament to its feature representation power.

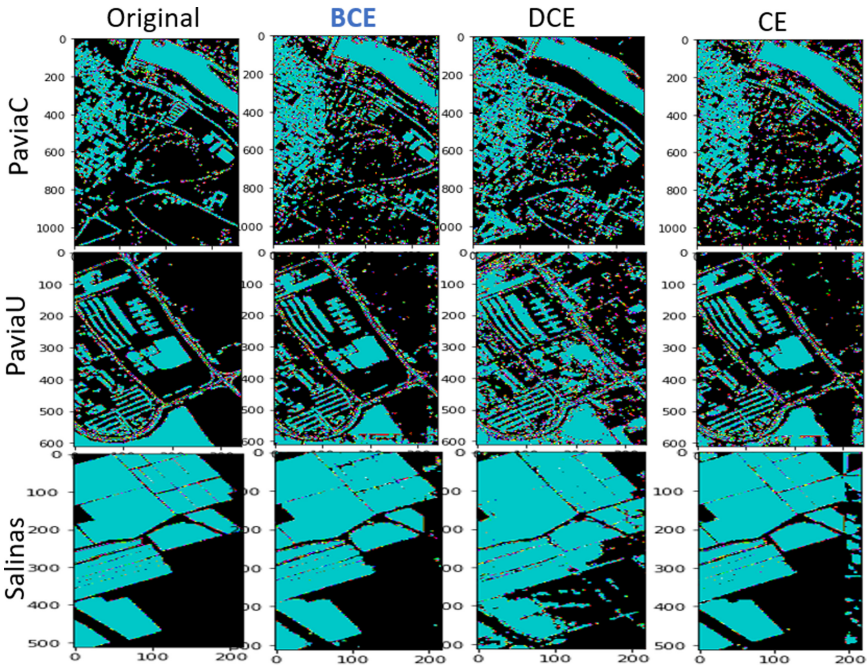

Fig. 6. Classification results for the three encoders on the datasets

4.6.3 Change Detection Results

The three encoders were concatenated with the change detection network and their-change detection map was compared for visual similarity and with the appropriate performance metrics (accuracy, recall, precision, and F1-score). Figure 7 show the bitemporal change detection map for the Hermiston city dataset and the results show that BCE has the smallest extra artifacts and CE having the lowest performance in this case. To get a better view a multiclass change detection map of the test patches is given on Fig. 8. There it can be clearly seen that BCE is almost identical with the original change map, while the others have some minor extra artifacts. BCE's feature representation is able to better represent the between class separability information in the feature space.

Table 4. Classification results summary via performance metrics

Data sets	Performance evaluation metric	Dimensionality reduction models		
		CE	DCE	BCE
PaviaC (10 classes)	Accuracy	0.36	0.88	**0.93**
	Recall	0.66	0.44	**0.69**
	Precision	0.67	0.45	**0.75**
	F1-score	0.63	**0.70**	**0.70**
PaviaU (10 classes)	Accuracy	0.92	0.83	**0.94**
	Recall	0.82	0.59	**0.87**
	Precision	0.80	0.69	**0.88**
	F1-score	0.81	0.57	**0.88**
Salinas (17 classes)	Accuracy	0.94	0.65	**0.96**
	Recall	**0.96**	0.76	0.94
	Precision	0.91	0.77	**0.94**
	F1-score	0.93	0.70	**0.94**

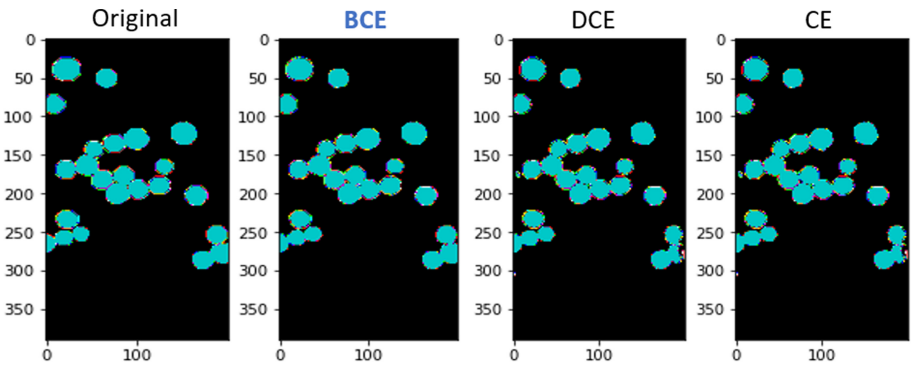

Fig. 7. Change detection shape map for Hermiston city dataset

A numerical analysis of the change detection performance is given on Table and Figure. Also, a confusion matrix is given for the three models to better visuality the rate of classification and misclassification. Table 5 summary shows that BCE significantly outperforms the others on all measure with the exception of F1-Score where it ranks close to the DCE.

Table 5. Change detection results via performance evaluation metrics

Dataset	Models		
	CE	DCE	BCE
Hermiston City (Accuracy)	0.96804	0.98626	**0.99372**
Hermiston City (Recall)	0.61131	0.76273	**0.87152**
Hermiston City (Precision)	0.70575	0.78573	**0.96203**
Hermiston City (F1-Score)	0.79516	**0.93854**	0.88978

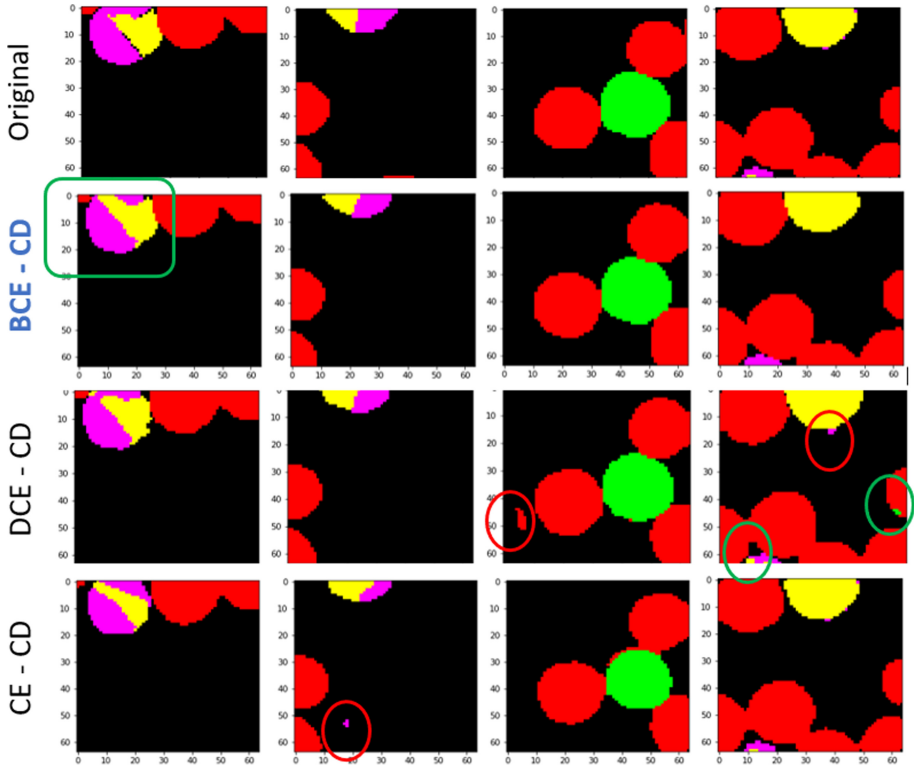

Fig. 8. Multiclass change detection map with extra artifacts highlighted

A graphical representation on Fig. 10 also shows a better visual comparison of the performance metrics (Fig. 9).

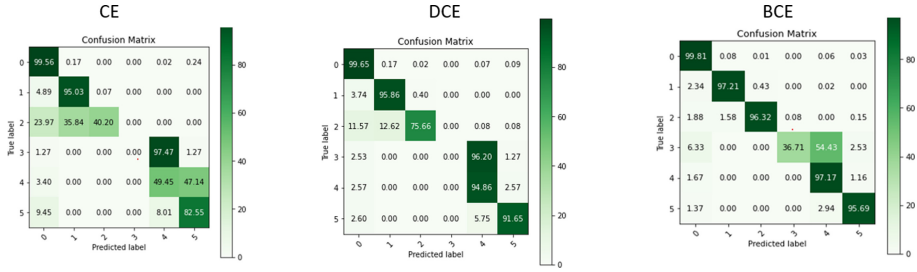

Fig. 9. Confusion matrix for the multiclass change detection map

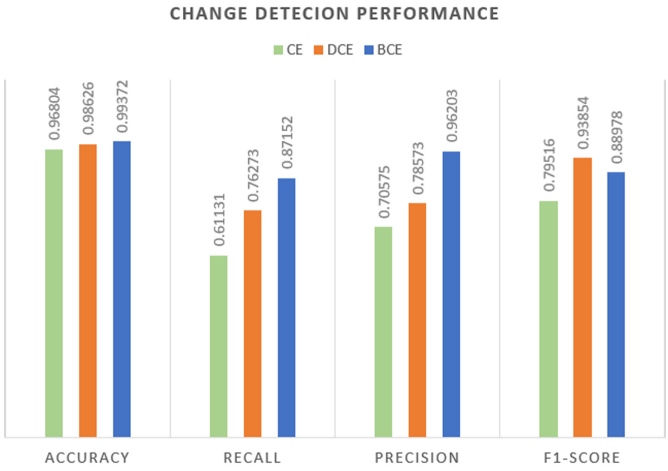

Fig. 10. Graphical summary of the change detection performance metrics

5 Conclusion

Hyperspectral image classification and change detection are important applications of hyperspectral images for remote sensing and other fields. One of the main challenges when performing either classification or change detection on hyperspectral images is the high dimensional nature of the images due to the large number of spectral bands present. Overall, the research was designed to explore other autoencoder techniques for the reduction of hyperspectral images. Namely a convolutional autoencoder and a densely connected autoencoder network have been selected as a comparative technique to compare the proposed techniques performance.

The branching autoencoder network has a better reconstruction capacity as demonstrated by the visual appearance of the reconstructed images and reconstruction loss being the smallest out of the three. For the classification task the dimensionality reduction models were integrated with a classifier to be trained in and to end manner. Out of the three models the classification result using the BCE section had better visual consistency and performance with different metrics (overall accuracy, precision, recall, and F1-score). Finally, the change detection model was trained using the Hermiston bitemporal images in supervised manner. The change map prediction showed that the model integrated with the BCE has fewer artifacts and overlaps on the different classes that the other two models.

References

1. Graña, M., Veganzons, M., Ayerdi, B.: Hyperspectral Remote Sensing Scenes. http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes
2. Manolakis, D.G., Lockwood, R.B., Cooley, T.W.: Hyperspectral Imaging Remote Sensing: Physics, Sensors, and Algorithms. Cambridge University Press, Cambridge (2016)

3. Asokan, A., Anitha, J.: Change detection techniques for remote sensing applications: a survey. Earth Sci. Inf. **12**(2), 143–160 (2019). https://doi.org/10.1007/s12145-019-00380-5
4. Shukla, A., Kot, R.: An overview of hyperspectral remote sensing and its applications in various disciplines. IRA-Int. J. Appl. Sci. (ISSN 2455-4499) **5**(2), 85 (2016). https://doi.org/10.21013/jas.v5.n2.p4
5. Liu, S., Marinelli, D., Bruzzone, L., Bovolo, F.: A review of change detection in multitemporal hyperspectral images: current techniques, applications, and challenges. IEEE Geosci. Remote Sens. Mag. **7**(2), 140–158 (2019). https://doi.org/10.1109/MGRS.2019.2898520
6. Sorzano, C.O.S., Vargas, J., Pascual-Montano, A.: A survey of dimensionality reduction techniques (2014)
7. Nguyen, L., Holmes, S.: Ten quick tips for effective dimensionality reduction. PLOS Comput. Biol. **15**(6), e1006907 (2019). https://doi.org/10.1371/journal.pcbi.1006907
8. Theodoridis, S., Koutroumbas, K.: Feature generation I: data transformation and dimensionality reduction. In: Pattern Recognition. Elsevier (2009)
9. Xie, H., Li, J., Xue, H.: A survey of dimensionality reduction techniques based on random projection (2017). http://arxiv.org/abs/1706.04371
10. Meyer-Baese, A., Schmid, V.: Feature selection and extraction. In: Pattern Recognition and Signal Analysis in Medical Imaging. Elsevier (2014)
11. Windrim, L., Ramakrishnan, R., Melkumyan, A., Murphy, R., Chlingaryan, A.: Unsupervised feature-learning for hyperspectral data with autoencoders. Remote Sens. **11**(7), 864 (2019). https://doi.org/10.3390/rs11070864
12. Ayma, V.H., Ayma, V.A., Gutierrez, J.: Dimensionality reduction via an orthogonal autoencoder approach for hyperspectral image classification. Int. Arch. the Photogram. Remote Sens. Spat. Inf. Sci. **XLIII-B3-2020**, 357–362 (2020). https://doi.org/10.5194/isprs-archives-XLIII-B3-2020-357-2020
13. Paul, A., Chaki, N.: Dimensionality reduction of hyperspectral images using pooling. Pattern Recogn. Image Anal. **29**(1), 72–78 (2019). https://doi.org/10.1134/S1054661819010085
14. Madhumitha Ramamurthy, Y., Harold Robinson, S., Vimal, A.: Auto encoder based dimensionality reduction and classification using convolutional neural networks for hyperspectral images. Microprocess. Microsyst. **79**, 103280 (2020). https://doi.org/10.1016/j.micpro.2020.103280
15. Mei, S., Ji, J., Geng, Y., Zhang, Z., Li, X., Du, Q.: Unsupervised spatial-spectral feature learning by 3D convolutional autoencoder for hyperspectral classification. IEEE Trans. Geosci. Remote Sens. **57**(9), 6808–6820 (2019). https://doi.org/10.1109/TGRS.2019.2908756
16. Song, A., Choi, J., Han, Y., Kim, Y.: Change detection in hyperspectral images using recurrent 3D fully convolutional networks. Remote Sens. **10**(11), 2018 (1827). https://doi.org/10.3390/rs10111827