HYPERSPECTRAL IMAGE CLASSIFICATION BY EXPLOITING CONVOLUTIONAL NEURAL NETWORKS

B. Hosseiny[1], H. Rastiveis[1, *], S. Daneshtalab[1]

[1] School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran - (ben.hosseiny, hrasti, somaye.daneshtalab) @ut.ac.ir

Commission III, WG III/4

KEY WORDS: Hyperspectral Image, Deep learning, Convolutional Neural Networks, Feature extraction

ABSTRACT:

High spectral dimensionality of hyperspectral images makes them useful data resources for earth observation in many remote sensing applications. In this case, the convolutional neural network (CNN) can help to extract deep and robust features from hyperspectral images. The main goal of this paper is to use deep learning concept to extract deep features from hyperspectral datasets to achieve better classification results. In this study, after pre-processing step, data is fed to a CNN in order to extract deep features. Extracted features are then imported in a multi-layer perceptron (MLP) network as our selected classifier. Obtained classification accuracies, based on training sample size, vary from 94.3 to 97.17% and 92.35 to 98.14% for Salinas and Pavia datasets, respectively. These results expressed more than 10% improvements compared to the classic MLP classification technique.

1. INTRODUCTION

Hyperspectral images mostly contain hundreds of spectral bands that can make a continuous spectral signature for every observed pixel (Chang, 2003). This property makes them useful for earth observation, and remote sensing tasks such as image classification, anomaly detection, and target detection. However, high dimensional data needs a high amount of labeled samples to obtain reliable results. This problem, known as the Hughes phenomenon, may cause redundancies and disturbances (Hughes, 1968; Yu et al., 2017). Therefore, the classification of hyperspectral images has always been an important and challenging problem in remote sensing communities.

In the last two decades, various classification algorithms, specifically based on machine learning techniques, have been proposed for hyperspectral image classification. Recently, the state-of-the-art deep-learning (DL) based methods have contributed a great improvement (Zhang et al., 2016). For example, Chen et al., (2014) employed stacked autoencoder containing five layers as deep architecture to a hyperspectral image transformed by Principal Component Analysis (PCA) in order to extract deep features to be fed in Support Vector Machine (SVM) classifier. Also, Yue et al., (2015) proposed a CNN-based spectral-spatial classifier for hyperspectral data classification after transforming the data using the PCA in order to cancel data redundancy and dimension reduction. The transformed image was fed to a four-layer CNN and logistic regression network to classify each pixel of the image. In (Hu et al., 2015), the authors trained a feed-forward neural network with five layers including an input layer, one-dimensional (1-D) convolutional layer, a max-pooling layer, a fully connected layer, and an output layer.

Handling the overfitting problem in hyperspectral image classification has been investigated in many studies. For instance, (Chen et al., 2016) proposed a deep CNN architecture containing L2 regularization terms and dropouts to avoid overfitting. In another study, Chen et al., (2017) combined Gabor filtering with CNN to converge faster and avoid overfitting. Limited training samples in relation to high parameters of CNN was also studied in (Yu et al., 2017) by considering data augmentation, appropriate convolutional kernel size, larger drop rates in the dropout layers, discarding the most commonly used max-pooling layers and fully connected layers. Another DL-based classification method named RPNet (Random Patches Network) was designed by Xu et al., (2018). In their network, convolutional kernels are selected randomly from input patches without any training. Recently, a novel framework called multiple convolutional layers fusion, which aims to fuse extracted information from different convolutional layers for HSI classification, was proposed by Zhao et al., (2019). Moreover, the multiscale convolution and diversified metric to obtain discriminative features for hyperspectral image classification was developed by Gong et al., (2019). Their CNN consists a multiscale filter bank, a concatenate layer to combine these multiscale features, and a fully connected layer to extract global features.

In this paper, we are going to investigate the classification of hyperspectral imagery by exploiting convolutional neural networks as a deep feature extractor. The remainder of this paper is organized as follows: Section 2 explains the theoretical background, and the methodology utilized in the paper. Section 3 describes the investigated datasets. Experimental results and discussions are gathered in section 4, and finally, the conclusions of the paper are presented in section 5.

2. METHODOLOGY

Figure 1 shows the framework of the proposed method for hyperspectral image classification. As can be seen from this figure, the proposed method includes data pre-processing, feature extraction, and classification. The pre-processing step consists of normalization, denoising, and dimensionality reduction. In the feature extraction step, a number of useful descriptors are extracted from the input data and are then fed to the classifier step in order to discriminate different classes and generate the classified map. More details of each step are described in the following subsections.

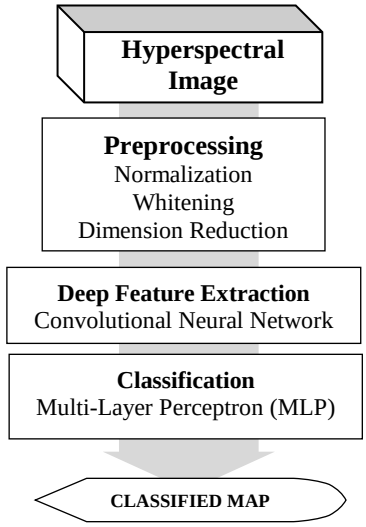


Figure 1. The proposed framework for classification of hyperspectral images.

2.1 Data Pre-processing

Raw input data usually includes noise, disturbance, and redundancy that needs to be fixed before entering to the main processing steps. Also, input layers may have a different scale and dynamic ranges that can affect other layers. Inherently scaling and whitening the input data, PCA can transform the input data into a space that data has the largest variations in every axis (Theodoridis and Koutroumbas, 2009). As a result, in our proposed classification method, after normalization of the input data, the PCA transformation is also performed.

2.2 CNN as Deep Feature Extractor

Complex scattering mechanisms, atmospheric effects, intraclass variabilities, and low signal-to-noise ratio (SNR) in hyperspectral images may deform the spectral characteristics of the object of interest, and make it difficult to extract effective features (Chen et al., 2016). Therefore, in the proposed framework, CNN is used as a deep feature extractor to extract high-level and robust features. A typical layer of a convolutional network consists of three stages. In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations. Second, each linear activation is run through a nonlinear activation function such as the rectified linear activation function. This stage is sometimes called the detector stage. In the third stage, a pooling function is used to modify the output of the layer (Goodfellow et al., 2016; Hu et al., 2015).

The input cube-data is decomposed to a 3D patch of dimensions $r \times c \times d$, where r and c correspond to the rows and columns of the patch and d to its depth (number of input bands). Each of these patches contains spectral and spatial information of centered pixel and its neighbors. Each patch is fed to network and after convolving with filters at every convolutional layer, deep features are eventually extracted in the final layer. These features are then fed to the MLP classifier which is responsible for the classification task.

2.3. Multi-layer perceptron classification

The last step after feature extraction is to classify the feature map into labels of interest. The MLP is a suitable choice in solving classification problems especially in the case of using deep features. In this case, deep feature extractor and the classifier can be stacked to generate an integrated system for classification. MLP generally consists of three main parts. The first part is the input layer that gets a vector containing features of interest. The second part involves hidden layers with plenty of fully connected neurons and activation function to apply non-linearity to linear matrix multiplications. The third phase is the output layer that usually includes neurons equal to the number of labels. Each neuron of the output layer represents a class label and represents a probability value. The winner neuron or the selected label for an input feature vector would be the neuron with the highest probability (Theodoridis and Koutroumbas, 2009).

3. DATASET

In this research, the proposed method is tested on two famous datasets. The first one is Salinas dataset collected by the 224-band AVIRIS sensor over Salinas Valley, California, and is characterized by high spatial resolution (3.7-meter pixels). The water absorption bands, in this case, bands: [108-112], [154-167], 224 are discarded. Salinas ground truth contains 16 classes. The color composite of this image and the corresponding ground truth data are shown in Figure 2-a and 2-b, respectively.

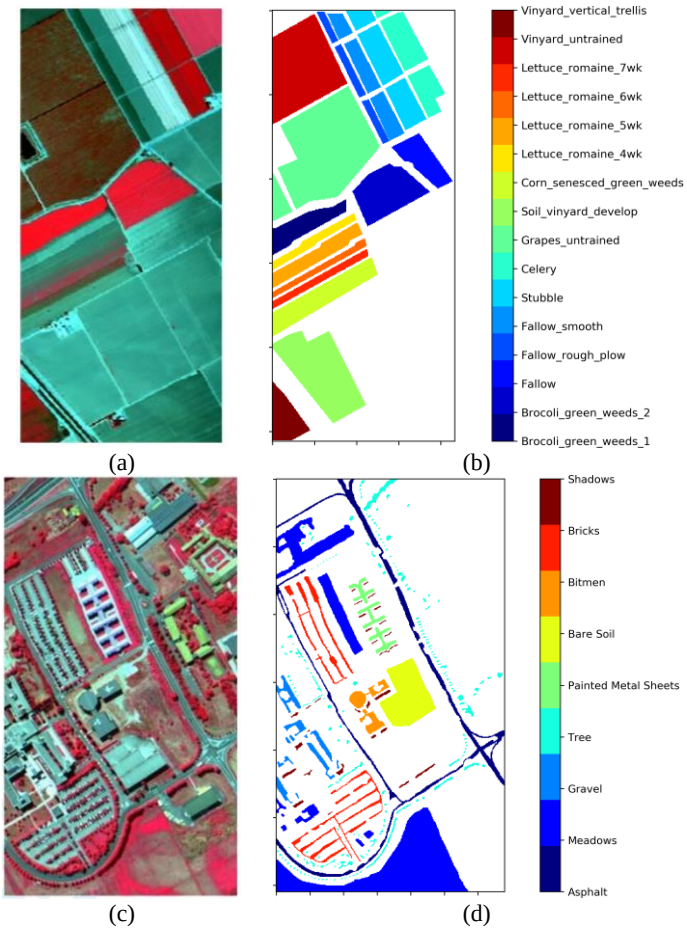


Figure 2. Salinas and PaviaU datasets (a) False color of Salinas (b) Ground-Truth of Salinas (c) False color of PaviaU (d) Ground-Truth of PaviaU

The second dataset, named Pavia University (PaviaU), was captured over an urban area surrounding the University of Pavia, Italy, and was recorded by the ROSIS-03 sensor. The image is of size 610×340×115 with a spatial resolution of 1.3 m per pixel, and spectral coverage ranging from 0.43 to 0.86 m. In the experiments, 12 noisy bands have been removed and the remaining 103 spectral channels were used for classification. Nine classes of interest are considered in the ground truth of this image. The color composite of this data and the corresponding ground truth the in Figure 2-c and 2-d.

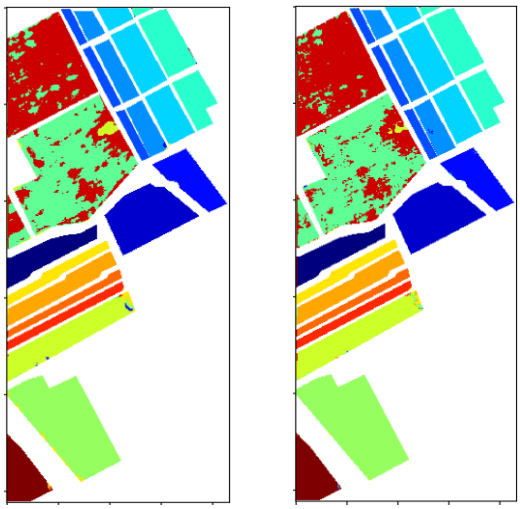
4. RESULTS AND DISCUSSION

After preparing input data by normalizing every layer and PCA transform at the pre-processing step, different network configurations on the mentioned datasets were performed. Mini-batch Adam optimizer (Kingma and Ba, 2014) was used with a learning rate of 0.00005 and batch-size of 50. Categorical cross-entropy was used as loss function in order to calculate classification error in every epoch. Also, 100 training samples were used to train neural networks. Input data to network is the first ten bands of PCA-transformed hyperspectral image that include more than 99.9% of the information of the original data. In this paper, different architectures based on the number of convolutional and fully-connected layers are investigated. For this purpose, in single-layered scenarios such as using only one convolutional layer, 100 filters are implemented, but when using two convolutional layers, the number of filters implemented for each layer is 50. Similar to that, for fully-connected section of the network, 100 neurons are used in single-layered scenarios and in double-layered scenarios, 50 neurons are implemented in each hidden layer. Also, in order to improve the network's stability and generalization 10% dropout and batch normalization were considered. ReLU activation function was used in every layer to increase nonlinearity of the generated model. Convolutional filters are 3×3 filters, and outputs were zero-padded after every convolutional layer to keep data patches at the fixed dimensions. Table 1 summarizes the constant parameters that are used in the implemented networks.

Optimizer	Adam
Learning rate	0.00005
Mini-batch size	50
Loss function	Categorical Cross-Entropy
Epochs	<1000
Training Samples	100
Feature conditioning	0.999
Dropout	10%
Conv. Filters	100 (50+50)
FC Neurons	100 (50+50)
Activation Function	ReLU
Kernel size	3×3
Padding	same

Table 1. Constant parameters used for investigating network performance

Figure 3 shows the visual classification results of the Salinas dataset using simple MLP with one and two hidden layers. In this case, 100 random samples per each class, are selected and used as the training dataset.



(a) (b)

Figure 3. Classification results of the Salinas with simple MLP containing (a) one hidden layer (b) two hidden layers

Various combinations of the convolutional and fully-connected layers with different input patch sizes were tested in order to find the optimal classifier network architecture. Three input patch sizes with dimensions of 3×3, 5×5, 7×7 were tested. Also, number of the tested convolutional and fully-connected layers vary between 1-2 and 0-2, respectively.

Figure 4 shows the visual classification results of the Salinas dataset based on the explained combinations. In this case, 100 random samples per each class, are selected and used as the training dataset. Also, quantitative results of the conducted experiments are gathered in Table 2. Every cell of this table is the obtained overall accuracy of the Salinas dataset for a specific experimented classifier network.

Exploiting 2-D convolutional layers makes the extraction of deep spatial features possible. Therefore, As Table 2 indicates, CNN has made about 10% improvement in the classification accuracy. Also, using bigger patch-size helps to extract more spatial features. However, the main drawback of the larger input patch size is increasing the number of trainable parameters of the network which cause much computational cost. Moreover, it can be noticed that the best result was obtained when the input patch-size is 7×7 and the network contains two convolutional and two fully-connected layers. Figure 5 shows the obtained optimum architecture in details.

Patch Size	CNN layers	Fully-Connected Layers					
		0		1		2	
		Train (%)	Test (%)	Train (%)	Test (%)	Train (%)	Test (%)
P=3	1	96.64	91.22	98.83	92.11	97.58	91.85
	2	96.56	90.49	97.27	90.15	98.12	92.20
P=5	1	96.72	90.82	99.22	92.61	97.5	91.71
	2	97.58	91.62	98.67	92.32	98.91	92.54
P=7	1	98.91	92.07	99.30	92.40	99.38	92.44
	2	99.06	91.86	99.45	92.80	98.91	**94.3**
MLP				86.16	81.99	89.92	83.18

Table 2. Obtained classification results of the Salinas dataset different network configurations

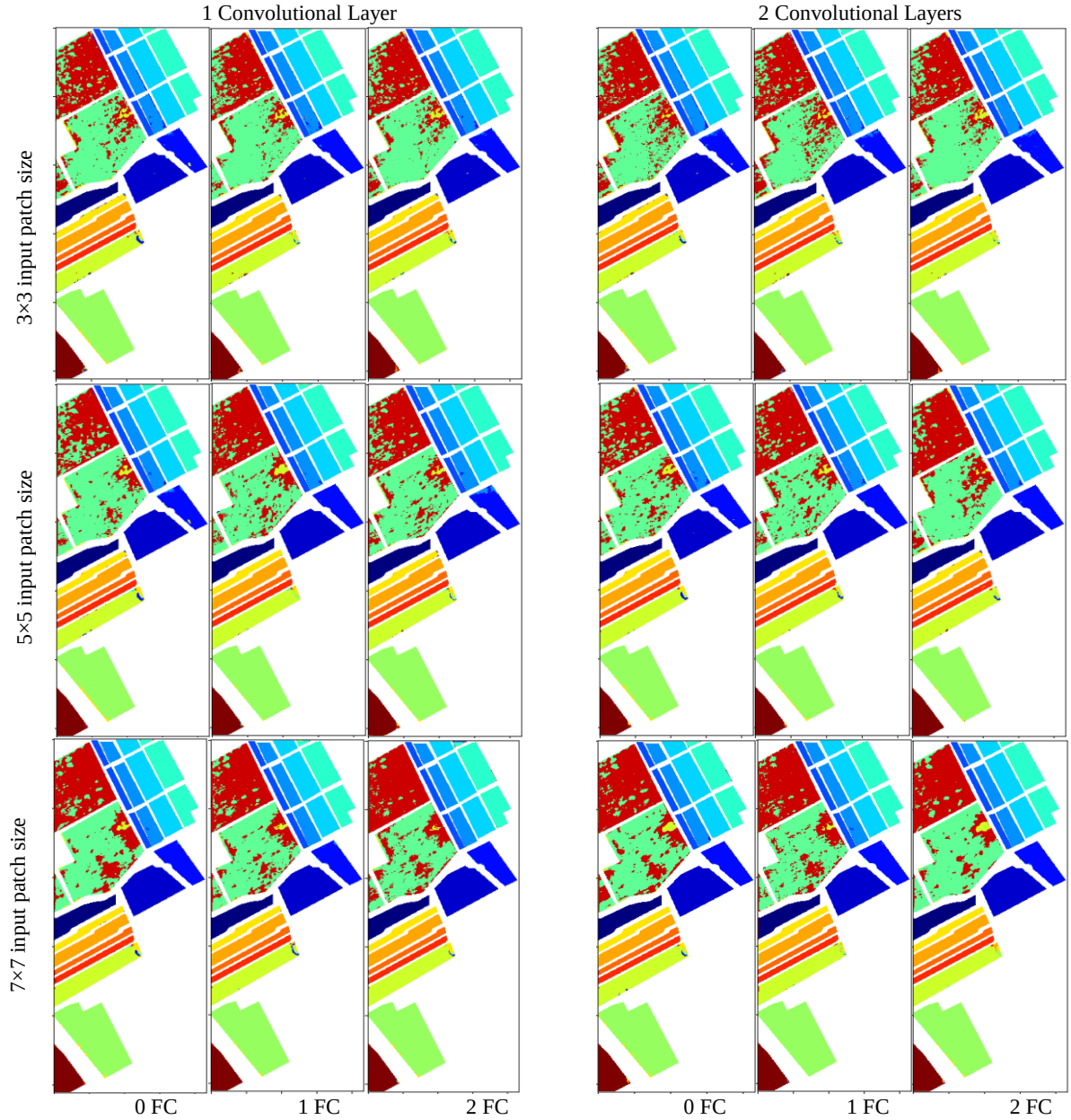


Figure 4. Classification results of the Salinas dataset by 2-D convolutional feature extraction with different input patch sizes.

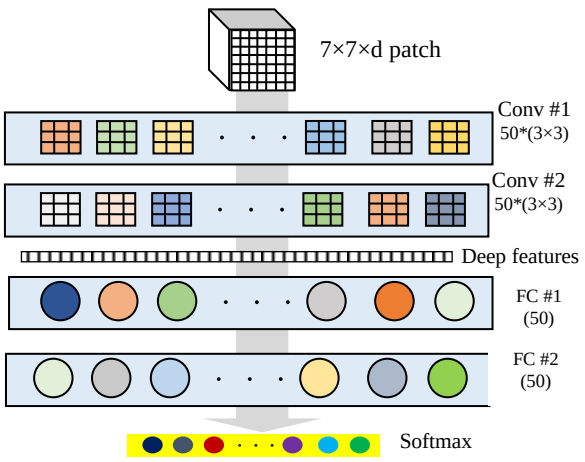
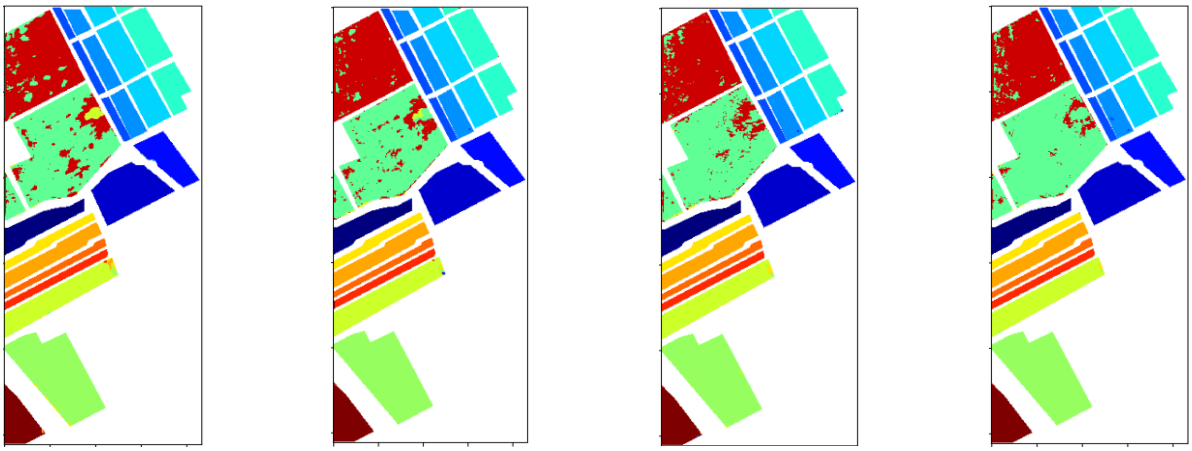


Figure 5. Architecture of the optimum network based on the results of Table 2.

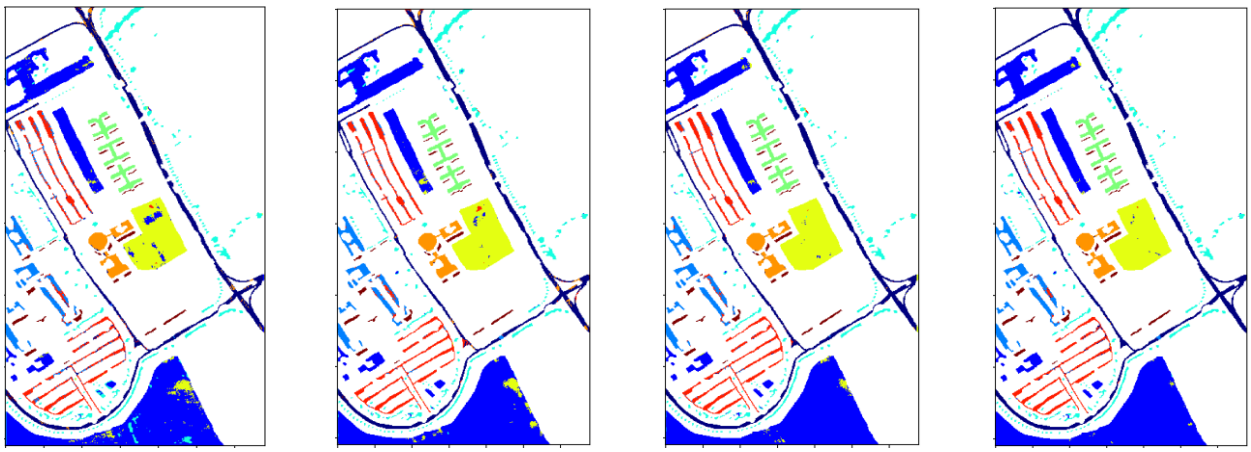
The obtained optimum network was also tested with various training samples. Figure 6 shows the classified maps of the Salinas dataset, obtained by training the optimum network with 100, 200, 300, and 400 samples of each class. The evaluation results of each classified map are gathered in Table 3. It can be seen there is direct relation between the accuracy and the number of training sample. In other words, the higher the number of training data, the higher the accuracy obtained. In which, the overall accuracy of the test data has increased from 94.3% to 97.17% by changing the number of training samples from 100 to 400.

The optimum selected network was also trained on PaviaU dataset. The classified maps and accuracy assessment of the various training samples for this dataset are shown in Figure 7 and Table 4. Based on these results, our proposed network has an overall accuracy of 98.14% when 400 samples of every class were used to train the network.



(a) (b) (c) (d)

Figure 6. Classification results of Salinas with 7×7 input patch with 2 convolutional layers and 2 fully-connected layers trained with (a) 100 training samples (b) 200 training samples (c) 300 training samples (d) 400 training samples



(a) (b) (c) (d)

Figure 7. Classification results of Pavia with 7×7 input patch with 2 convolutional layers and 2 fully-connected layers trained with (a) 100 training samples (b) 200 training samples (c) 300 training samples (d) 400 training samples

Training Samples	Train OA (%)	Test OA (%)
100	98.91	94.3
200	98.84	96.06
300	99.74	96.23
400	99.92	97.17

Table 3. Results obtained for classification of Salinas after training different train samples

Training Samples	Train OA (%)	Test OA (%)
100	100	92.35
200	98.75	95.28
300	100	97.38
400	100	98.14

Table 4. Results obtained for classification of PaviaU after training different train samples

5. CONCLUSION

In this paper, convolutional neural networks were exploited in order to extract deep features from a hyperspectral image. The main process includes three steps: preprocessing, feature extraction, and classification. Different architectures of CNNs were investigated in order to extract deep features and obtain better classification results. In the end, extracted features were fed to an MLP network to classify input data. Salinas and PaviaU as two famous datasets of hyperspectral images were used in our experiments. By comparing the experimental results of CNN based classification architectures with simple MLP, 10-15% of improvements in overall accuracy can be found.

REFERENCES

Chang, C.-I., 2003. Hyperspectral imaging: techniques for spectral detection and classification. Springer Science & Business Media.

Chen, Y., Jiang, H., Li, C., Jia, X., Ghamisi, P., 2016. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. IEEE Trans. Geosci. Remote Sens. 54, 6232–6251.

Chen, Y., Lin, Z., Zhao, X., Wang, G., Gu, Y., 2014. Deep learning-based classification of hyperspectral data. IEEE J. Sel. Top. Appl. earth Obs. Remote Sens. 7, 2094–2107.

Chen, Y., Zhu, L., Ghamisi, P., Jia, X., Li, G., Tang, L., 2017. Hyperspectral Images Classification With Gabor Filtering and Convolutional Neural Network. IEEE Geosci. Remote Sens. Lett. 14, 2355–2359.

https://doi.org/10.1109/LGRS.2017.2764915

Gong, Z., Zhong, P., Yu, Y., Hu, W., Li, S., 2019. A CNN With Multiscale Convolution and Diversified Metric for Hyperspectral Image Classification. IEEE Trans. Geosci. Remote Sens. 1–20. https://doi.org/10.1109/TGRS.2018.2886022

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep learning. MIT press.

Hu, W., Huang, Y., Wei, L., Zhang, F., Li, H., 2015. Deep convolutional neural networks for hyperspectral image classification. J. Sensors 2015.

Hughes, G., 1968. On the mean accuracy of statistical pattern recognizers. IEEE Trans. Inf. Theory 14, 55–63. https://doi.org/10.1109/TIT.1968.1054102

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv Prepr. arXiv1412.6980.

Makantasis, K., Karantzalos, K., Doulamis, A., Doulamis, N., 2015. Deep supervised learning for hyperspectral data classification through convolutional neural networks, in: 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE, pp. 4959–4962.

Theodoridis, S., Koutroumbas, K., 2009. Pattern recognition. Academic Press.

Xu, Y., Du, B., Zhang, F., Zhang, L., 2018. Hyperspectral image classification via a random patches network. ISPRS J. Photogramm. Remote Sens. 142, 344–357. https://doi.org/10.1016/J.ISPRSJPRS.2018.05.014

Yu, S., Jia, S., Xu, C., 2017. Convolutional neural networks for hyperspectral image classification. Neurocomputing 219, 88–98.

Yue, J., Zhao, W., Mao, S., Liu, H., 2015. Spectral–spatial classification of hyperspectral images using deep convolutional neural networks. Remote Sens. Lett. 6, 468–477. https://doi.org/10.1080/2150704X.2015.1047045

Zhang, L., Zhang, L., Du, B., 2016. Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art. IEEE Geosci. Remote Sens. Mag. 4, 22–40. https://doi.org/10.1109/MGRS.2016.2540798

Zhao, G., Liu, G., Fang, L., Tu, B., Ghamisi, P., 2019. Multiple convolutional layers fusion framework for hyperspectral image classification. Neurocomputing 339, 149–160. https://doi.org/10.1016/J.NEUCOM.2019.02.019