Hyperspectral Image Classification using Semi-supervised Deep Learning Strategies

Sourish Gunesh Dhekane[1], Shivam Tiwari[1], and Manan Sharma[1]

[1] Indian Institute of Information Technology Guwahati, India
{sourishdhekane,shivam21ballia,manansharma858}@gmail.com

Abstract Recent development in deep learning (DL) methodologies have shown promising results on various computer vision tasks including classification of hyperspectral data. However, these methodologies are expected to suffer in the presence of lack of training data, due to complex network architecture and a large number of parameters. In this paper, various K-means based clustering techniques are explored to generate pseudo labels to facilitate the training of deep networks. To tackle the curse of dimensionality, an auto-encoder (AE) based dimensionality reduction method is employed. Finally, the classification is done using Convolutional Long Short Term Memory Cells (ConvLSTM) which outperforms the rest of the deep neural networks used. In addition, an analysis of the effect of the proposed dimensionality reduction method on classification accuracy is presented. The efficacy of the proposed approach is demonstrated on two real-world hyperspectral image data sets namely the "University of Pavia" (UP) and "Salinas".

Keywords: Hyperspectral Image Classification, Semi-supervised Learning, Auto-encoder, Convolutional Long Short Term Memory Cells.

1 Introduction

A hyperspectral image can be denoted by a three-dimensional tensor (a, b, λ) where a and b denote the spatial coordinates and λ denotes the wavelength associated with the spectral bands. One of the key aspects of hyperspectral images is the presence of these large number of narrow, contiguous spectral bands [1] containing vast amount of information. This information can be used in various remote sensing applications like land cover classification [2], crop protection and analysis [3], and mineral detection [4]. In most of the applications, the task of classifying pixels of a hyperspectral image plays a key role. However, there are a number of challenges associated with this task. One of the biggest challenges is the lack of correctly labeled pixels (ground truth knowledge), which directly affects the quality of classification maps [5]. Also, the shortage of labeled pixels results in relatively less amount of training data compared to the spectral dimensions of the hyperspectral image. This introduces the curse of dimensionality in this application. Also, it has been observed that for a constant set of training data, the classification performance increases with increase in distinctive features (dimensions) until an optimal number is reached [6]. As this optimal number

is crossed, the performance of the classifier starts to degrade which is exactly what happens in case of this problem statement. The spectral dimensions of the hyperspectral image are large in number as well as redundant in nature. Hence, extracting idiosyncratic features from the raw input data in order to reduce the dimensionality plays an important role in the classification task.

In recent years, due to the availability of massive amounts of data and the improvement in computational resources, deep learning [7] based methodologies are gaining success in various computer vision applications like image classification [8], semantic segmentation [9], and estimation in generative models [10]. In the case of classification of hyperspectral images in a supervised scenario, Roy et al. [11] have achieved the state-of-the-art results using a Hybrid Spectral Convolutional Neural Network (HybridSN). However, the same task in a semi-supervised scenario demands some modifications in the deep learning methodologies due to the aforementioned challenges. To incorporate these modifications, Liu et al. [12] proposed a semi-supervised Convolutional Neural Network (CNN) containing skip connection parameters between encoder and decoder layer. On similar grounds, a semi-supervised ladder network [13] was proposed that jointly optimized a supervised as well as an unsupervised cost. In another work, Pan et al. proposed a Multi-grained Network (MugNet) [14] based on a multi-grained scanning approach where kernels are generated in a semi-supervised way. These approaches try to modify either the network architecture with its associated operations or the optimization problem. However, a relatively simple approach to deal with the scarcity of labeled samples is to generate pseudo labels using a clustering approach on the abundant unlabeled data and a few available ground truth samples [15]. The generated pseudo labels can then be used to train the deep neural network. Following this approach, a Convolutional Recurrent Neural Network (CRNN) was used for classification by treating each pixel as a sequence of spectral dimensions [16]. It used pseudo labels generated by clustering based on Constrained Dirichlet Process Mixture Model (C-DPMM) to pre-train the proposed model. The fine-tuning was done using the limited labeled data. In this work, we further explore this approach by testing the performance of K-means based clustering algorithms for pseudo label generation. In the case of dimensionality reduction, AE have demonstrated good performance in encoding the high dimensional redundant input into a low-dimensional manifold [17]. Although traditional approaches for dimensionality reduction like PCA are good in the removal of correlated features, it is biased towards features having large variance which leads to incorrect results. Hence, it is intuitive to check the efficacy of deep AE for this task.

In this paper, we explore the approach of pseudo label generation via different conventional techniques based on semi-supervised version of K-means clustering algorithm to facilitate the pre-training of the model. Although CRNN has shown better results in this approach [16], recurrent neural networks (RNNs) are known to suffer from the vanishing and exploding gradient problem [18]. Long Short Term Memory (LSTM) cells overcome this drawback by introducing ‘update, forget, and output gates’ in the architecture. Hence, we explore Convolutional Long Short Term Memory Cells (ConvLSTM) in the aforesaid semi-supervised setting. Also, we explore AE to extract better representations from the raw feature set. Finally, we fine-

tune the pre-trained architecture using the limited labeled samples and compare the performance of our approach with other classifiers in the same setting.

The remaining paper is organized into 4 sections. Section 2 contains the detailed description of the clustering methods, the dimensionality reduction algorithm, and the classifier that are chosen to be analyzed in this work. Section 3 contains the description of the proposed methodology. Section 4 covers the description of datasets used, experimental setups, and the results. Finally, Section 5 provides an analysis of the obtained results with conclusions and the scope of future work.

2 Related Work

2.1 Clustering Approaches

Clustering is a task of partitioning the given unlabeled data into different groups (clusters) such that the data samples in the same cluster have relatively high similarity with each other as compared to data samples from different clusters. Clustering algorithms are unsupervised i.e. information about the class labels cannot be extracted from the clusters. However, when the label information about a few data samples is available, it is possible to make an approximation to relate the clusters with class labels [19]. This approach is known as Semi-supervised clustering and the class labels predicted by this approximation are called Pseudo labels [15]. K-means clustering is one of the most widely used clustering algorithms in machine learning literature. In the following sub-sections, we describe the semi-supervised versions of K-means clustering and its variants: K-median and K-medoid based K-means clustering.

K-means clustering. It is an iterative hard clustering approach which groups the data into K clusters [20]. As K is a hyper-parameter in this algorithm, the very first step is to specify the value of K. Then, K number of cluster centers are initialized which are selected randomly from the set of data points. Next, the distance of every data point from each cluster center is calculated in terms of a chosen distance metric. The data points are assigned to that cluster center to which they are the closest. After this process is completed, the position of cluster centers is updated by calculating the mean of the data points assigned to their respective cluster. These steps are carried until convergence where the updates in cluster centers are sufficiently small. In other words, the K-means clustering algorithm aims to minimize the objective function J denoted in equation (1), where x_j^i represents assignment of data point x_j to the i^{th} cluster with its center located at c_i.

$$J = \sum_{i=1}^{k} \sum_{j=1}^{n} \left\| x_j^i - c_i \right\|^2 \tag{1}$$

One way to make a semi-supervised version of this algorithm is to initialize the cluster centers at data samples for which the label information is provided. Suppose there are c number of classes present and the label information about l number of randomly selected data samples per class are provided. Then, K should be set equal to $l * c$.

After the initialization of $l * c$ cluster centers, the assignment of data samples to cluster centers and the updation of the cluster centers is carried out as described before. Finally, each data sample is assigned a pseudo label which is the class label of the cluster center to which the data sample is assigned. Hence, for each iteration, a unique pseudo label is assigned to the data samples. The only assumption made in this semi-supervised version of K-means clustering algorithm is the availability of label information of at least one data sample per class. This assumption is reasonable for the task at hand.

Although K-means clustering is a relatively simple and scalable algorithm that guarantees convergence, it is affected adversely by the presence of outliers. To avoid this issue, different variants of K-means clustering, like K-medoid [21] and K-median [22] clustering, are considered.

K-medoid based K-means clustering (K-med). The main difference between the K-medoid and K-means clustering algorithm lies in the assignment of cluster centers. In K-means clustering, the cluster center is not necessarily a data sample. However, K-medoid clustering imposes an additional condition that the cluster centers can only be located at data samples. Hence, it is more robust to the presence of outliers. However, traditional K-medoid clustering algorithm becomes computationally expensive as the number of data samples increase. This is due to a step in K-medoid clustering algorithm where each medoid is swapped with every non-medoid data sample and the corresponding cost is calculated. If the cost in the new set of medoids is lesser than the previous set of medoids, then the new set is used for further iterations. In case of a large number of data samples, this implementation is infeasible due to a large number of swapping operations. This issue can be resolved by applying the approach of K-medoid clustering in K-means clustering. In K-med clustering algorithm, the cluster center is not updated to the mean of the data samples assigned to a particular cluster. Instead, it is updated to the data sample which is nearest to this calculated mean. This solution ensures that the cluster center is always a data sample.

K-median clustering. The K-median clustering algorithm is designed to reduce the cost associated with clustering which is calculated using the L1 norm. Hence, the objective function to be minimized for K-median can be written as in equation (2) where x_j^i represents assignment of data point x_j to the i^{th} cluster with its center located at c_i. The reduction in the cost of clustering is done by assigning the cluster center to the median instead of the mean of the data samples belonging to a particular cluster.

$$J = \sum_{i=1}^{k} \sum_{j=1}^{n} |x_j^i - c_i| \tag{2}$$

The K-median clustering algorithm does not impose the condition that a cluster center must be a data sample from the dataset. However, the median is calculated per feature using L1 distance. Thus, the values of individual dimensions of the cluster center come from the dataset.

2.2 Auto-encoder

Traditional Autoencoder consists of two main components: the Encoder and the Decoder [23]. The task of the encoder is to map the input feature vector into an intermediate hidden representation. When the objective of the AE is to minimize the dimensionality of the data, the number of neurons in the hidden layer is set to be less than that of the input layer. Hence, the encoder embeds input into a low-dimensional manifold. The task of the decoder is to reconstruct the original input from its low dimensional encoding. The working of AE can be mathematically represented for input X as shown in the set of equations (3) where e is the encoded representation obtained by the set of weights w and bias vector b and X' is the reconstructed output features. Here, J denotes the squared loss which is minimized by the AE.

$$\begin{aligned} e &= f(wX + b) \\ X' &= g(w'e + b') \\ J(X, X') &= \|X - X'\|^2 \end{aligned} \tag{3}$$

In order to obtain a low-dimensional embedding, the number of hidden layer units should be considerably less than the number of features in the input. Another way to improve the AE is to introduce sparsity by restricting the number of active neurons in the hidden layer for any input [24]. This is done by including a penalty term in the objective function which maintains the value of sparsity parameter. Hence, AE is a good choice to reduce dimensionality and learn useful representations in the presence of a large number of redundant features.

2.3 Convolutional Long Short Term Memory Cells (ConvLSTM)

In the literature, the task of hyperspectral image classification has been achieved using deep neural networks like Convolutional Neural Networks (CNN) [27], Recurrent Neural Networks (RNNs) [28], Convolutional Recurrent Neural Networks (CRNN) [16], etc. CNNs are better at extracting the features based on information based on spatial relations, whereas RNNs are good at predictions on sequential data. Although it is not directly apparent that hyperspectral data is sequential, due to the high number of dimensions, a pixel of a hyperspectral image can be viewed as a sequence of spectral dimensions. Hence, RNN can be expected to perform well. However, for better understanding of the long-range dependencies in sequential data, LSTMs have been demonstrated to perform better than the RNNs [18]. The LSTM architecture consists of a memory cell and a number of gates having their own parameters. The input gets collected to the memory cell in case of activation of the Input Gate. Preservation of long term context is the key aspect of LSTM. This is achieved by the Forget Gate. In case of activation of the Forget Gate, the state of the past cell is removed (forgotten). In this way, only important contexts are preserved. Also, the decision on whether the cell output will be propagated to the final state is done by the Output Gate. Due to this architecture, the vanishing/exploding gradient problem is addressed [25].

However, due to the use of fully connected layers, spatial information cannot be encoded in the LSTM framework [26]. Hence, the introduction of convolution layers instead of fully connected layers in input-to-state and state-to-state transitions in LSTM can help in the incorporation of information about the spatial relations. The mathematical representation of ConvLSTM is shown in the set of equations (4). The memory cell mem_t of the LSTM is the space where the information of the current state is stored. The input gate is represented by inp_t in which the information about the new input is accumulated. The forget gate frg_t decides whether the information about the past state should be forwarded into the upcoming states. The output gate out_t decides whether the recent output of the mem_t is passed to the final state h_t. It should be noted that the symbol $*$ represents the convolution operation and the symbol o represents the hadamard product.

$$\begin{aligned} inp_t &= \sigma(w_{xi} * x_t + w_{hi} * h_{t-1} + w_{ci} \, o \, mem_{t-1} + b_i) \\ frg_t &= \sigma(w_{xf} * x_t + w_{hf} * h_{t-1} + w_{cf} \, o \, mem_{t-1} + b_f) \\ out_t &= \sigma(w_{xo} * x_t + w_{ho} * h_{t-1} + w_{co} \, o \, mem_t + b_o) \\ mem_t &= frg_t \, o \, mem_{t-1} + inp_t \, o \, tanh(w_{xc} * x_t + w_{hc} * h_{t-1} + b_c) \\ h_t &= out_t \, o \, tanh(mem_t) \end{aligned} \tag{4}$$

Due to the aforementioned architecture, the ConvLSTM is better at predictions on sequential data in addition to the consideration of information based on spatial relations.

3 Proposed Methodology

The proposed semi-supervised deep learning framework consists of 4 steps. The first step consists of the use of semi-supervised clustering in order to generate pseudo labels. The second step extracts meaningful features and reduces the dimensionality of the hyperspectral image using AE. In the third step, the pseudo labels and extracted features are used to pre-train the ConvLSTM. Lastly, the ConvLSTM is fine-tuned using the same set of labeled samples used for semi-supervised clustering. The structure of the proposed methodology is shown in Fig. 1.

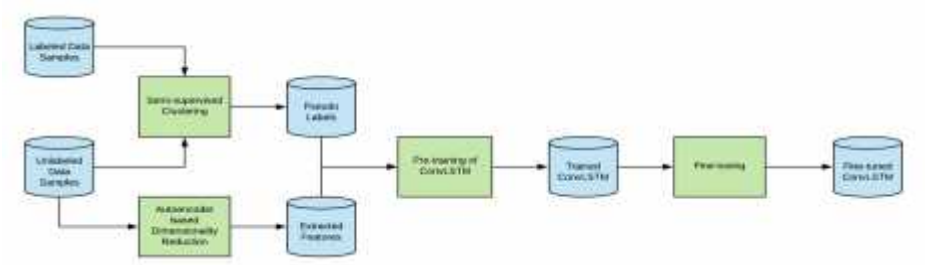

Fig. 1. The proposed semi-supervised deep learning strategy

3.1 Pseudo Label Generation

Pseudo labels are the class labels predicted based on the cluster assignment obtained by a clustering algorithm along with a small set of labeled data samples. The semi-supervised versions of K-means, K-med, and K-median clustering algorithm are explored in this work to generate the pseudo labels. The data samples for which the label information is available are assigned as the initial cluster centers. This is followed by the execution of aforesaid clustering algorithms. Finally, pseudo labels are assigned as described in Section 2.1.

3.2 Dimensionality Reduction

The set of unlabeled data is used to train the AE in an unsupervised manner. As the motive behind the use of AE is to reduce the dimensionality, the number of neurons in the hidden layer is set to be less than that of the number of neurons in the input layer. Sparsity regularizer is also introduced in the cost function in order to learn a better encoding. It should be noted that no label information is used in this step i.e. fine-tuning of AE by stacking a softmax layer is avoided. The quality of extracted features is measured by using them to train the ConvLSTM along with the generated pseudo labels. The analysis of information loss due to the reduction of dimensionality is carried out by decreasing the dimensions of the data by approximately 10% of the total dimensions in a step-wise manner and checking the classification accuracy obtained against it.

3.3 Pre-training the ConvLSTM

The pseudo labels generated are used to pre-train the ConvLSTM which takes into input the features extracted by the AE. In this step, it should be noted that the proposed model is trained on the pseudo labels, which might not be fully accurate when compared with the ground truth. As the training and test data is partitioned randomly, the training data is more likely to contain some falsely predicted pseudo labels along with the majority of correctly predicted pseudo labels. To improve the performance of the classifier in this step, one way is to increase the number of correctly predicted pseudo labels by improving the accuracy of pseudo label generation. The other way is to ensure that less number of incorrectly predicted pseudo labels are used for training. This can be done by introducing a metric of confidence in the pseudo label generation step. For example, if a data sample P1 is more close to the cluster center to which it is assigned than another data sample P2 which is assigned to the same cluster center, then P1 is more likely to be having the class label of that cluster center than that of P2. Hence P1 should more likely to be included in the training set than P2. Thus P1 has more confidence in the generated pseudo label as compared to P2. If this metric of confidence is used in the implementation, then the training data is selectively chosen to consist of data samples with the pseudo labels having relatively high confidence. Otherwise, train and test data is divided in a random fashion.

3.4 Fine Tuning the ConvLSTM

The full utilization of the given labeled data can be achieved by fine-tuning the pre-trained ConvLSTM with the labeled data. It is expected that the performance of the classifier should improve with the training on this labeled set. The fine-tuning step can be done in a number of ways. A set of dense layers can be added to the pre-trained model followed by a softmax layer as proposed in [16] where the parameters of the pre-trained model are set to be frozen during the fine-tuning. Hence, only the parameters corresponding to the newly added dense and softmax layer are changed. However, we observe that the addition of a large number of dense layers to the pre-trained model decreases the performance. Hence, we propose to include only a single dense layer and a softmax layer to the pre-trained model.

4 Experiments and Results

4.1 Datasets

Two real-world hyperspectral image datasets are used to check the performance of the proposed methodology.

The first dataset used is called the “University of Pavia” (UP) dataset [29]. The dataset was captured using Reflective Optics System Imaging Spectrometer (ROSIS) sensors during a flight campaign over the targeted geographical region. The original dataset is a 610*340 pixel hyperspectral image. However, many pixels contain no information. Hence, they are discarded before the analysis. The spatial resolution of the image is 1.3 meters. It contains 103 spectral dimensions within a narrow wavelength range of 430-860 nanometers. This dataset consists of 9 different land cover classes such that the classes are not represented equally i.e. the data set does not have equal number of data samples in each class. Hence, the issue of class imbalance is present.

The second dataset used is called “Salinas” dataset [30]. This dataset is captured using Airborne Visible / Infrared Imaging Spectrometer (AVIRIS) sensors. It is collected over the Salinas Valley in California. The spatial resolution of this image is 3.7 meters, which is higher as compared to UP dataset. It is a 512*217 pixel image containing at-sensor radiance data. It consists of 224 spectral dimensions. However, 20 dimensions out of the total 224 dimensions are discarded as they correspond to the water absorption bands. The ground truth of this dataset contains 16 different classes and similar to the UP dataset, the classes are not represented equally. Thus, the issue of class imbalance is present in this dataset also.

4.2 Experimental Setup

In the experiment, the labeled set is constructed by randomly selecting 10, 20, 30, and 40 pixels per class. In the semi-supervised clustering step, K-means, K-mediod, and K-median clustering algorithms are tested. We define a configuration by the clustering algorithm and the number of labeled samples used in the semi-supervised cluster-

ing. Hence, for 3 different clustering algorithms and 4 settings of the number of labeled samples used, there are 12 configurations in total. For each of the configuration, the same set of 10 / 20 / 30 / 40 labeled samples is used to remove the effect of initialization of cluster centers on the performance of pseudo label generation. The number of iterations for which the clustering should be conducted is set as a hyper-parameter. As the proposed methodology requires only an approximation of the class labels in the form of pseudo labels, the number of iterations is set to be a very small number. This implies that the clustering is forcefully terminated after a few iterations. This also helps in achieving less execution time.

To analyze the information loss in the process of feature extraction, the spectral dimension is reduced in steps of 10% of the total, till it reaches half of the total number of dimensions. For example, in case of UP dataset, the classification accuracy is obtained for the feature sets with 93, 83, 73, 63, and 53 dimensions (reduction of 10 dimensions in each step). The total number of dimensions of the UP dataset is 103. Finally, the effect of dimensionality reduction is analyzed by plotting a graph between the cardinality of the reduced feature set and the classification accuracy associated with it. The classification accuracies in this experiment are obtained by using ConvLSTM as the classifier over the reduced feature set.

Finally, the performance of the proposed model is compared with 3 different classifiers: CNN, LSTM, and CNN-LSTM. For CNN, the feature vector is passed to 1-dimensional convolution layer with 64 filters, each having kernel size of 8. Padding is applied to keep the feature shape unchanged. ReLU activation and a dropout of 0.2 is used in all neural architectures. Then, a batch normalization layer is applied followed by a 1-dimensional max pooling layer of stride 2. The training of the architecture is carried out with ADAM optimizer. The LSTM model is implemented using an LSTM layer having the number of units equal to the size of input feature vector. Both the dropout and recurrent dropout are set to 0.2. The training of the architecture is carried out with ADAM optimizer. In the case of CNN-LSTM, a 1-dimensional convolution layer having 64 filters of size 8 is used with the 'SAME' padding and ReLU activation. A max pooling layer is attached to it with stride 4. It is followed by an LSTM layer having number of units equal to the size of the original feature vector. Finally, a dense layer is applied with a softmax layer attached to it. The model is trained with "Adam" optimizer. In the implementation of ConvLSTM, instead of passing the 1-dimensional feature vector for a pixel, a 2-dimensional feature vector is constructed. To incorporate the spatial information, the spectral dimensions of the neighbor pixels are concatenated and passed to the 2-dimensional ConvLSTM layer. The output of this layer is passed to a dense layer attached to a softmax layer. It should be noted that the number of units in the ConvLSTM layer is set to be equal to the number of spectral features of the input. Each of the classifiers is tested 10 times for each of the aforesaid 12 configurations. The average of all these results are reported.

4.3 Results

The classification accuracies for the UP and the Salinas dataset are shown in Table.2 and Table.3. The accuracy of the best performing classifier in each configuration is

highlighted. The index numbers 1, 2, 3, and 4 represent the classifiers CNN, LSTM, CNN-LSTM, and ConvLSTM respectively. It can be seen that for a classifier, the classification accuracy increases with the increase in the number of labeled samples used per class. Also, the use of K-med and K-median clustering algorithm has shown better performance on the UP dataset. However, the use of K-means has outperformed the other two clustering algorithms in case of the Salinas dataset. From the point of view of the classifiers, ConvLSTM has produced the best results for a majority of the configurations with a few exceptions, where CNN-LSTM can be seen to outperform the ConvLSTM. An important observation in this experiment is the improved results due to the combined use of spatial and sequential information (CNN-LSTM and ConvLSTM) as compared to its use separately (CNN and LSTM).

Table 1. Classification accuracies for UP dataset

	K-means				K-med				K-median			
No.	10	20	30	40	10	20	30	40	10	20	30	40
1	88.76	89.31	90.03	92.57	90.08	92.31	93.28	94.86	89.93	90.91	92.33	94.30
2	88.12	90.63	91.32	92.08	89.80	90.22	90.87	91.21	88.12	90.97	91.41	92.55
3	90.44	92.12	93.47	94.07	91.32	**93.62**	**93.86**	94.01	90.94	**93.23**	93.84	94.71
4	**91.32**	**93.43**	**94.01**	**95.30**	**92.03**	92.77	93.54	**95.61**	**91.83**	92.96	**94.14**	**95.42**

Table 2. Classification accuracies for Salinas dataset

	K-means				K-med				K-median			
No.	10	20	30	40	10	20	30	40	10	20	30	40
1	93.47	94.81	95.84	96.03	93.58	94.27	94.32	94.56	92.34	93.06	93.41	94.66
2	91.82	91.99	92.08	92.25	91.03	91.39	91.68	91.90	90.72	90.95	91.63	92.00
3	**93.78**	**95.91**	96.26	96.88	93.08	**94.44**	95.09	96.13	**93.21**	93.98	94.91	**96.26**
4	93.69	95.53	**96.44**	**96.97**	**93.25**	94.29	**95.38**	**96.52**	93.14	**94.33**	**95.12**	96.05

The classification accuracies obtained on the extracted features are shown in Fig. 2. The results clearly highlight the fact that the amount of information loss in the feature extraction is comparatively lesser than that of the reduction in dimensions. This shows the efficacy of the AE to extract distinguishing features from the large set of raw input features. Thus, from the mentioned results and the4.4 observations stated above, it can be clearly seen that the proposed methodology and the choice of classifier outperforms other classifiers in the same environment in the majority of configurations.

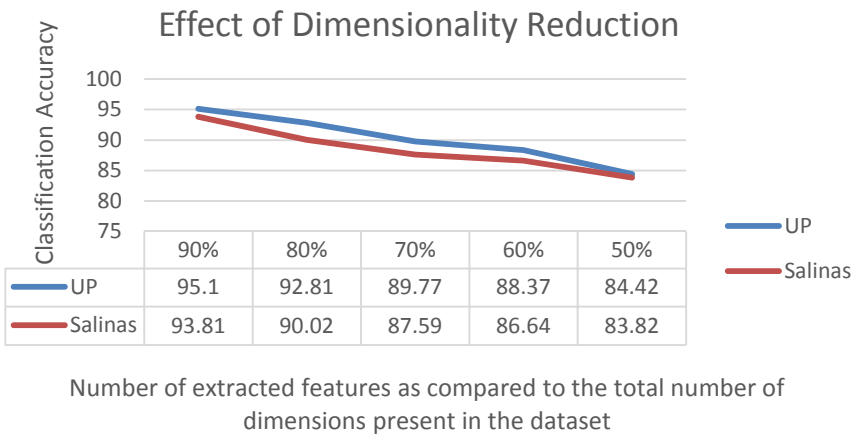

Fig. 2. Classification accuracies using the extracted features

5 Conclusion and Future Work

This paper explores the semi-supervised deep learning based strategies to classify hyperspectral images. The key aspect of this methodology is the generation of more accurate pseudo labels via semi-supervised clustering. The highest overall accuracies of 95.61% and 96.97% are obtained for the UP dataset and Salinas dataset respectively. Also, relatively low information loss is observed which is marked by only 10% reduction in overall accuracy due to the dimensionality reduction. These two aspects highlight the effectiveness of the proposed methodology. Apart from these two measures, the proposed methodology is also computationally efficient due to early termination of the semi-supervised clustering algorithm. However, this methodology can be further explored as the pseudo label generation step can be performed by several other more efficient clustering algorithms.

Acknowledgments

The authors thank Prof. Paolo Gamba for providing the UP dataset.

References

1. Chang, C. I.: Hyperspectral imaging: techniques for spectral detection and classification Vol. 1, Springer Science & Business Media. (2003).
2. Xu, B., Gong, P.: Land-use/land-cover classification with multispectral and hyperspectral EO-1 data. Photogrammetric Engineering & Remote Sensing, 73(8), pp. 955-965. (2007).
3. Mahlein, A. K., Oerke, E. C., Steiner, U., Dehne, H. W.: Recent advances in sensing plant diseases for precision crop protection. European Journal of Plant Pathology, 133(1), pp. 197-209. (2012).

4. Schmidt, F., Legendre, M., Le Mouëlic, S.: Minerals detection for hyperspectral images using adapted linear unmixing: LinMin. Icarus, 237, pp. 61-74. (2014).
5. Baraldi, A., Bruzzone, L., Blonda, P.: Quality assessment of classification and cluster maps without ground truth knowledge. IEEE Transactions on Geoscience and Remote Sensing, 43(4), pp. 857-873. (2005).
6. Houghes, G. F.: On the mean accuracy of statistical pattern recognition. IEEE Trans. Inform. Theory, 14(1), pp. 55-63. (1968).
7. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning nature, 521(7553), 436 (2015).
8. Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097-1105. (2012).
9. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587. (2014).
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Bengio, Y.: Generative adversarial nets. In Advances in neural information processing systems pp. 2672-2680. (2014).
11. Roy, S. K., Krishna, G., Dubey, S. R., Chaudhuri, B. B.: HybridSN: Exploring 3-D-2-D CNN Feature Hierarchy for Hyperspectral Image Classification. IEEE Geoscience and Remote Sensing Letters (2019).
12. Liu, B., Yu, X., Zhang, P., Tan, X., Yu, A., Xue, Z.: A semi-supervised convolutional neural network for hyperspectral image classification. Remote Sensing Letters, 8(9), pp. 839-848. (2017).
13. Büchel, J., Ersoy, O.: Ladder Networks for Semi-Supervised Hyperspectral Image Classification. arXiv preprint arXiv:1812.01222. (2018).
14. Pan, B., Shi, Z., Xu, X.: MugNet: Deep learning for hyperspectral image classification using limited samples. ISPRS Journal of Photogrammetry and Remote Sensing, 145, pp. 108-119. (2018).
15. Wu, H., Prasad, S.: Semi-supervised dimensionality reduction of hyperspectral imagery using pseudo-labels. Pattern Recognition, 74, pp. 212-224. (2018).
16. Wu, H., Prasad, S.: Semi-supervised deep learning using pseudo labels for hyperspectral image classification. IEEE Transactions on Image Processing, 27(3), pp. 1259-1270. (2017).
17. Wang, Y., Yao, H., Zhao, S.: Auto-encoder based dimensionality reduction. Neurocomputing, 184, pp. 232-242. (2016).
18. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In International conference on machine learning, pp. 1310-1318. (2013).
19. Bair, E.: Semi- supervised clustering methods. Wiley Interdisciplinary Reviews: Computational Statistics, 5(5), pp. 349-361. (2013)
20. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability Vol. 1, No. 14, pp. 281-297. (1967).
21. Kaufman, L., P. J. Rousseeuw, and Y. Dodge.: Clustering by Means of Medoids in Statistical Data Analysis Based on the. pp. 405-416. (1987).
22. Jain, A. K., Dubes, R. C.: Algorithms for clustering data Vol. 6. Englewood Cliffs, NJ: Prentice hall. (1988).
23. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press. (2016).
24. Makhzani, A., Frey, B.: K-sparse autoencoders. arXiv preprint arXiv:1312.5663. (2013).

25. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation, 9(8), pp. 1735-1780. (1997).
26. Xingjian, S. H. I., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., Woo, W. C.: Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Advances in neural information processing systems, pp. 802-810. (2015).
27. Hu, W., Huang, Y., Wei, L., Zhang, F., Li, H.: Deep convolutional neural networks for hyperspectral image classification. Journal of Sensors, 2015. (2015).
28. Lyu, H., Lu, H., Mou, L.: Learning a transferable change rule from a recurrent neural network for land cover change detection. Remote Sensing, 8(6), 506. (2016).
29. Gamba, P. A collection of data for urban area characterization. In IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium (Vol. 1). (2004, September).
30. http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes