International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Special Issue 31(September, 2015), PP. 111-116

DEEP LEARNING METHOD FOR SATELLITE IMAGE CLASSIFICATION:A LITERATURE REVIEW

Komal M. Sukre,Imdad A. Rizvi and Mahesh M. Kadam
Department of Electronics & Telecommunication Engineering,
Terna Engineering College,Mumbai University,Nerul,Navi Mumbai,India.

Abstract- **Many traditional signal processing techniques and machine learning utilize shallow architectures which consist of a single layer of non-linear feature transformation. Examples of shallow models are nonlinear or linear dynamic models, conditional random models, maximum entropy models, markov hidden models, maximum entropy models, multilayer perceptron and kernel regression with only one hidden layer. A property mutual to these shallow architectures models are simple architecture which consists of only one layer responsible for altering the basic input signals into a problem specific feature space, which we can't observe. The deep learning paradigm tackles problems on which shallow architectures (e.g. SVM) are altered by the express of dimensionality. Some Part of a two stage method learning involving many layers of nonlinear processing a set of statistically substantial characteristics are automatically extracted from data. Deep learning method can be used in applications like remote sensing such as Land cover Classification, Detection of Vehicle in Satellite Images, Hyper spectral Image classification.**

Keywords—Deep learning, deep belief net, Auto Encoder Land cover classification, hyper-spectral data classification, Restricted Boltzmann Machines (RBMs), PolSAR

I. INTRODUCTION

One of the important applications in Geographic Information System is classification of Land Cover in urban area. It is necessary for many purposes such as urban landscape pattern analysis, urban land management, urban planning and disasters monitoring. Remotely sensed data is using for land cover mapping in urban area for decades. Within many remote sensing systems, Synthetic Aperture Radar (SAR) is recognized as a strong resource for urban analysis, as it's less influenced by solar radiance or conditions of weather in comparison to optical sensors. Since more sporadic information could be collected in Multi polarizations, polarimetric SAR data which is used for Land cover classification increasingly [1]. By combining imaging hyper spectral remote sensing data and spectroscopy technology can get spectrally and spatially continuous data. Hyper spectral data are becoming an important tool for observing the surface of Earth [2] [3] and these are used in a deep array of applications. A general technology in these applications is the each pixel classification in hyper spectral data. If successfully used, the hyper spectral data can produce high classification accuracies and more specified class assortments [4]. Generally, Land cover classification methods and in hyper spectral classification, there are two types of approaches Non parametric and Parametric approaches. Parametric approaches including Maximum Likelihood Classifier, Minimum Distance Classifier and Expectation Maximization Algorithm usually need proper assumptions for distribution of data. For Multitemporal or Multisource data, the distribution is however difficult to model. On the other hand, Non Parametric approaches like Decision Trees, Genetic Algorithms, Artificial Neural Networks and Support Vector Machines are widely used in both classification. However, the performances of the non-parametric approaches mainly depend on the selected features. As an advanced machine learning approach emerged in modern years, deep learning has been triumphingly applicable in the field of image classification. With deep architecture, approaches like Deep Belief Networks (DBN) [5] [6] can represent data with complex spatio-temporal statistical patterns. From above all approach, Deep Learning approach can automatically extract effective classification features, which is helpful for land cover mapping. For hyper spectral classification our work depends on applying Auto-Encoder which is based on deep architecture models, to learn deep features of hyper spectral data in an unsupervised method. Our methods exploit single layer Auto-Encoder and Multi-Layer stacked Auto-Encoder to learn shallow and deep features of hyper spectral data, respectively. In this paper, we acquaint deep learning-based feature extraction for hyper spectral image classification and land cover classification.

II. HYPERSPECTRAL DATA CLASSIFICATION

1. *Deep Learning*

Deep learning contains a class of layers which try to step by step learn deep features of input data with deep neural networks, typically more than three or four layers. The network is initialized first layer by layer via unsupervised training and then well-adjusted in a supervised manner. In this procedure, high level characteristics can be learned from low level characteristics, whereas the appropriate features can be developed for image classification in the end. According to some recent papers [7] [8], deep architecture can give better approximation than shallow architecture. Deep neural network architectures consist of deep belief networks [9], deep Boltzmann machines [10], Auto-Encoders [11], and stacked Auto-Encoder[12].

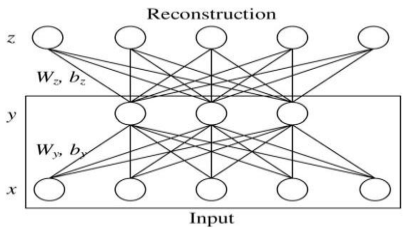

Fig .1 Single layer AE for hyper-spectral data classification [2].

The architecture determines a hidden feature “y” from input “x” by reconstructing it on “z”. Related parameters are showed in the network [16]. The training models are layer-wise which have a group of alternatives such as restricted Boltzmann machines [13], pooling units [14], convolutional neural networks [15], Auto-Encoders and denoising Auto-Encoders [11]. In this paper, we use one of the above deep learning architecture; Auto-Encoders used for hyper spectral data classification and choose Stacked Auto-Encoders as the corresponding deep architecture.

1.1 Auto-Encoders:

Auto-Encoders consist of visible layer of inputs, one layer of units which is hidden, one reconstruction layer and Fig 1 shows activation function. In training mapping of the input $x \in R^d$ to the hidden layer takes place first and produces the output $y \in R^h$. The network related to this step is shown in Fig. 1 and is called an “encoder.” After that mapping is done by a “decoder”. A “decoder” produces an output layer. Size of output layer is same as input layer. Output layer is called as “reconstruction.” The values in reconstructed layer are denoted as $z \in R^d$. Mathematically, these two steps can be formulated as

$$y = f\left(W_y x + b_y\right) \quad (1)$$

$$z = f\left(W_z y + b_z\right) \quad (2)$$

Where W_y and w_z denote the input to hidden and the hidden to output weights, respectively, b_y and b_z denote the bias of hidden and output units, and $f(.)$ denotes the activation function. There are many alternatives for $f(.)$ such as sigmoid function, hyperbolic tangent, and rectified linear function. In our paper, the following constraint holds.

$$W_y = W_z^{'} = W \quad (3)$$

We have three groups of parameters remaining to learn: W, b_y, b_z

The goal of training is to minimize the “error” between input layer and reconstruction layer, i.e.

$$\arg\min_{w, b_y, b_z} \left[c(x,z)\right] \quad (4)$$

$c(x,z)$ stands for the “error,” which can be defined in many ways. Thus, the weight updating rule can be defined as (where η denotes learning rate)

$$W = W - \eta \frac{\partial \cos t(x,z)}{\partial W} \quad (5)$$

$$b_y = b_y - \eta \frac{\partial \cos t(x,z)}{\partial b_y} \quad (6)$$

$$b_z = b_z - \eta \frac{\partial \cos t(x,z)}{\partial b_z} \quad (7)$$

After training the network, the reconstruction layer with its parameters is removed and the learned feature lies in the hidden layer, which is used for classification. During reconstruction, it only uses the information in hidden layer, which is encoded as features from input. If the architecture can recover original input perfectly from output that means it keeps full data of the input. So, stacking the encoders trained in this manner minimizes information loss. At the meantime, they maintain abstract and invariant information in deeper feature. This is the main reason for choosing AE to extract deep features for hyper spectral data.

1.2 Stacked AE:

Stacked Auto-encoder can be formed by using multiple Auto-encoders together. If we stacked input and output layers of Auto-encoders layer by layer then we get "stacked Auto-encoder". In first a 0th layer & 1st layer is there. Mapping of 0th layer inputs to 1st layer feature in first layer. In training process every single Auto-encoder of stacked auto-encoder processed as above method. Every time training of subsequent layers of Auto-encoder takes place with the help of outputs of previous layer

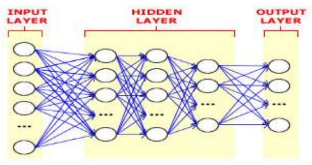

Fig .2 Instance of a SAE. It has five layers: one input layer, three hidden layers, and an output layer [16].

After this layer of training, the decoder of the third layer AE is useless and we are considering input to hidden parameters as weights between second & third layer

1.3 Restricted Boltzmann machine:

An RBM is a distinctive type of Markov random field consist of one layer of stochastic hidden units and one layer of stochastic visible or observable units. RBMs can be described as bipartite graphs as shown in Figure 3, In that graph all visible units are joined to all hidden units, and there are visible-visible or hidden-hidden connections not available.

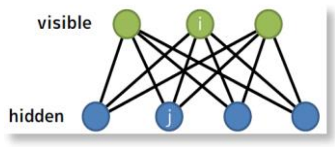

Fig .3 A RBM with *i* visible units and *j* hidden units [16].

In an RBM, the joint distribution $p(v,h;\theta)$ over the visible units v and hidden units h, given the model parameters u, is defined in terms of an energy function $E(v,h;\theta)$ of

$$p(v,h;\theta) = \frac{\exp(-E(v,h;\theta))}{z} \tag{8}$$

Where $z = \sum_v \sum_h \exp(-E(v,h;\theta))$ is a normalization factor, and the marginal probability that the model allots to a visible vector v is

$$p(v;\theta) = \sum_h \frac{\exp(-E(v,h;\theta))}{z} \tag{9}$$

Energy Function for a Bernoulli (visible)-Bernoulli (hidden) RBM, is defined as

$$E(v,h;\theta) = -\sum_{i=1}^{I}\sum_{j=1}^{J} W_{ij}v_ih_j - \sum_{i=1}^{I} b_iv_i - \sum_{j=1}^{J} a_jh_j \tag{10}$$

Where w_{ij} represents the symmetric communication term between visible unit v_i and hidden unit h_j, b_i and a_j are the bias terms, and I, *J are* the numbers of visible and hidden units. We can efficiently calculate the conditional probabilities as,

$$p(h_j=1|v;\theta) = \sigma\left(\sum_{i=1}^{I} w_{ij}v_i + a_j\right) \tag{11}$$

$$p(v_i|h;\theta) = \sigma\left(\sum_{j=1}^{J} w_{ij}h_j + b_i, 1\right) \tag{12}$$

Where $\sigma(x) = 1/(1+\exp(x))$. See a derivation in [25]. Where v_i takes real values and follows a Gaussian distribution with mean $\sum_{j=1}^{J} w_{ij}h_j + b_i$ and variance one. We can alter real value stochastic variables to binary stochastic variables by using Gaussian-Bernoulli RBMs, Further this can be processed using the Bernoulli-Bernoulli RBMs. Taking the gradient of the log likelihood log $p(v,\theta)$ we can obtain the regulation for the RBM weights as

$$\Delta w_{ij} = E_{data}(v_ih_j) - E_{model}(v_ih_j) \tag{13}$$

Where $E_{data}\left(v_i h_j\right)$ the expectation is observed in the training set and $E_{model}\left(v_i h_j\right)$ is that same expectation under the distribution defined by the model. Unfortunately $E_{model}\left(v_i h_j\right)$ is rough to compute that's why we used contrastive divergence (CD) approximation to the gradient. Where $E_{model}\left(v_i h_j\right)$ is replaced by running the Gibbs sampler initialized at the data for one full step [17]. Careful training of RBMs is essential to the success of applying deep learning to practical problems. A practical guide of the RBM training is provided in [18].

We have to train RBMs as a probabilistic model; the log-likelihood criterion is used to maximize. This can be achieved with gradient ascent from a training set D as follows:

$$\partial \log p\left(D\right) = \sum_{x\in D} \frac{\partial \log p(x)}{\partial w_{ij}}$$

$$= \sum_{x\in D} \frac{\sum_{g} \frac{\partial E(x,g)}{\partial w_{ij}}}{\sum_{g} e^{-E(x,g)}} - \sum_{x\in D} \frac{\sum_{u}\sum_{g} \frac{\partial E(u,g)}{\partial w_{ij}} e^{-E(u,g)}}{\sum_{u}\sum_{g} e^{-E(u,g)}}$$

$$= E_{data}\left[\frac{\partial E(x,g)}{\partial w_{ij}}\right] - E_{model}\left[\frac{\partial E(u,g)}{\partial w_{ij}}\right] \qquad (14)$$

Where the first term is the expectation of $\frac{\partial E(x,g)}{\partial w_{ij}}$ when the input variables are set to an input vector x and according to the conditional distribution p (h|x) the hidden units are sampled. The second term shows an expectation of $\frac{\partial E(u,g)}{\partial w_{ij}}$ when u and g are sampled similarly to the joint distribution of the RBM p (u,g) and is uncivilized,. It can be almost same as Gibbs sampling: starting from any configuration $\left(v^0, h^0\right)$ one samples h^t according to $p\left(h|v^{t-1}\right)$ and v^t according to $p\left(v|h^t\right)$ until the sample $p\left(v^t, h^t\right)$ is distributed nearly sufficient to the target distribution $p\left(v,h\right)$. In practice, We can reduced the number of step by starting the Markov chain with a sample from the training set of data and assuming that the model is not so far from target. This is the estimation on which the Contrastive Divergence (CD) learning algorithm is depends [19].

1.4 From stacked RBMs to deep belief networks:

In an RBM, the hidden units are conditionally independent to the visible units, but there is not statistically independency. Stacking RBMs goal is learning and analysing these dependencies with other RBM. In the stack visible layer of every RBM is situated to the hidden layer of the previous RBM (see Fig. 4). In under mentioned the deep learning scheme, First RBM can train from the input data and after that sequentially other RBMs are trained. Stacking RBMs rises a range on the log-likelihood [20], which helps the expectation to reform the working of the model by adding layers.

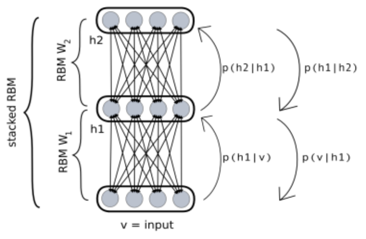

Fig .4 A stacked RBMs architecture [21].

A stacked RBMs architecture is a deep model. Patterns generated from the top RBM can be propagated back to the input layer using only the conditional probabilities as in a belief network. This setup is referred to as a Deep Belief Network [21].

III. DEEP BELIEF NETWORK

Stacking a number of the RBMs learned layer by layer from bottom-up gives rise to a DBN. The DBN (Deep Belief Network) model was introduced by Hinton *et al.* in 2006 [5] and it is one of the broadly explored and used deep learning architectures [22] [23]. The DBN is consisting of many layer neural networks made up of several stacked Restricted Boltzmann Machines. As the building blocks of the DBN, an RBM contains layer of visible units *v*, and a layer of hidden units *h*, connected by symmetrically weighted connections, as shown in Fig. 5.

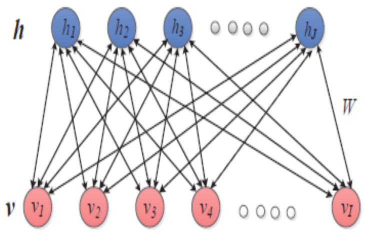

Fig .5 A Restricted Boltzmann Machine with *v* visible units and *j* hidden units [24].

Assuming binary units, the RBM defines the energy of joint configuration of the visible and hidden units (*v, h*) as

$$E(v,h;\theta) = -\sum_{i=1}^{I}\sum_{j=1}^{J} w_{ij} v_i h_j - \sum_{i=1}^{I} b_i v_i - \sum_{j=1}^{J} a_j h_j \qquad (15)$$

Where w_{ij} represents the symmetric communication term between visible unit v_i and hidden unit h_j, b_i and a_j are the bias terms, *I and J are* the numbers of visible and hidden units. The probability of an exclusive configuration of the visible and hidden units is proportional to the negative exponentiation. As the building blocks of the DBN, an RBM can be treated as hidden units are proportional to the negative exponentiation of the energy function

$$p(v,h) = e^{-E(v,h)} / \sum_{v}\sum_{h} e^{-E(v,h)} \qquad (16)$$

Given a visible layer vector *v* of the RBM, the probability that hidden node h_j is activated can be calculated as

$$p(h_j = 1|v) = \sigma\left(b_j + \sum_{i=1}^{I} v_i w_{ji}\right) \qquad (17)$$

where $\sigma(x) = 1/(1+\exp(x))$ the probability that visible layer node v_i is activated, given hidden layer vector *h*, can be calculated in a similar way as follows

$$p(v_i = 1|h) = \sigma\left(a_i + \sum_{j=1}^{J} h_j w_{ji}\right) \qquad (18)$$

RBM training process can be described as follows. When random initialization of the weights and biases is done, iterative training on the RBM will perform on the training data. Given the training data on the visible nodes, i.e. v_i , the hidden nodes states h_j are sampled according to Eq. (17). This is called the positive phase of RBM training. In the negative phase, a "reconstruction" of the visible nodes v_i is obtained according to Eq. (18). The positive phase is directed again to create $h_j^{'}$ after that, we can update RBM weights and biases by Contrastive-Divergence (CD) algorithm [24] through gradient ascent, as follows:

$$\Delta w_{ij} = \in \left(\langle v\, h_j\rangle - \langle v_i^{'} h_j^{'}\rangle\right) \qquad (19)$$

$$\Delta a_i = \in \left(\langle v_i\rangle - \langle v_i^{'}\rangle\right) \qquad (20)$$

$$\Delta b_j = \in \left(\langle h_j\rangle - \langle h_j^{'}\rangle\right) \qquad (21)$$

Where $\in$ Denotes the learning rate, and $\langle . \rangle$ refers to the expectation of the sampled states. The DBN takes a layer-wise learning strategy, in which RBMs are individually trained one after another in a bottom up fashion. For supervised classification, a softmax neuron network is placed on the top of the last RBM as a multiclass classifier. Therefore, the softmax classifier learns a joint model of the features extracted by the RBMs and the corresponding label of the samples. After training, the class label of a test sample can be predicted as the softmax's output by the forward propagation procedure in which the test data pass from the lowest level visible layer through multi RBM layers to the softmax output layer.

IV. CONCLUSION

We can classify hyper spectral image data. Hyper spectral deep features can be extracted by Stacked Auto Encoders (SAEs). It is shown that AE-extracted features are useful for classification, and it helps to increase the SVM accuracy and logistic regression while obtaining the highest accuracy when compared with other methods like PCA, KPCA, and NMF which are called as feature extraction method.

REFERENCES

[1] A. Moreira, P. Iraola, M. Younis, G. Krieger, I. Hajnsek, and K. Papathanassiou, "A tutorial on synthetic aperture radar," *IEEE Geoscience Remote Sensing Magazine*, vol. 1, no. 1, pp. 6–43, Mar. 2013.

[2] D. Landgrebe, “Hyperspectral image data analysis,” IEEE Signal Process. Mag., vol. 19, no. 1, pp. 17–28, Jan. 2002.

[3] J. A. Richards, Remote Sensing Digital Image Analysis: An Introduction. New York, NY, USA: Springer, 2013.

[4] S. Rajan, J. Ghosh, and M. M. Crawford, “An active learning approach to hyperspectral data classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 4, pp. 1231–1242, Apr. 2008.

[5] G. E. Hinton, S. Osindero, and Y. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[6] G. E. Hinton and R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, pp. 504–507, 2006.

[7] I. Sutskever and G. E. Hinton, “Deep, narrow sigmoid belief networks are universal approximators,” *Neural Comput.*, vol. 20, no. 11, pp. 2629–2636, Nov. 2008.

[8] N. LeRoux and Y. Bengio, “Deep belief networks are compact universal approximators,” *Neural Comput.*, vol. 22, no. 8, pp. 2192–2207, Aug. 2010.

[9] G. E. Hinton, S. Osindero, and Y. Teh, “A fast learning algorithm for deep belief nets*,” Neural Comput.,* vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[10] R. Salakhutdinov and G. E. Hinton, “Deep Boltzmann machines,” in *Proc. Int. Conf. Artif. Intell. Statist.*, Clearwater Beach, FL, USA, 2009, pp. 448–455.

[11] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks*,” in Proc. Neural Inf. Process. Syst.,* Cambridge, MA, USA, 2007, pp. 153–160.

[12] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, “Stacked denoising autoencoders,” *J.Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010

[13] G. E. Hinton, “Apractical guide to training restricted Boltzmann machines*,” Dept. Comput. Sci.,* Univ. Toronto, Toronto, ON, Canada, Tech. Rep. UTML TR2010-003, 2010.

[14] Y. LeCun et al., “Backpropagation applied to handwritten zip code recognition,” *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Apr. 1989.

[15] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980

[16] Yushi Chen, Zhouhan Lin, Xing Zhao, Gang Wang, and Yanfeng Gu,” Deep Learning-Based Classification of Hyperspectral Data”, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, VOL. 7, NO. 6, June 2014

[17] G. Hinton, S. Osindero, and Y. Teh, “A fast learning algorithm for deep belief nets,” *Neural Comput.*, vol. 18, pp. 1527–1554, 2006

[18] G. Hinton, “A practical guide to training restricted Boltzmann machines,” Univ. Toronto, Tech. Rep. 2010-003, Aug. 2010

[19] G. E. Hinton. “Training products of experts by minimizing contrastive divergence,”*Neur. Comput.*, 14:1771–1800, 2002.

[20] Y. Bengio, “Learning deep architectures for AI. Foundations and Trends in Machine Learning,” 2:1–127, 2009.

[21] G. E. Hinton, S. Osindero, and Yee-Whye ,”A fast learning algorithm for deep belief nets,” *Neur. Comput.*, 18:1527–1554, 2006.

[22] D. Yu and L. Deng, “Deep learning and its applications to signal and information processing,” *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 145–154, 2011.

[23] I. Arel, C. Rose, and T. Karnowski, “Deep machine learning-a new frontier in artificial intelligence research,” *IEEE Computational Intelligence Magazine*, vol. 5, no. 4, pp. 13–18, 2010.

[24] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[25] *Qi Lv*1*_, Yong Dou*1*, Xin Niu*1*, Jiaqing Xu*1*, Baoliang Li*12, ”Classification of land cover based on deep belief networks using polarimetric radarsat-2 data”, *National Laboratory for Parallel and Distributed Processing, School of Computer,* pp.4679-4682,IGARSS 2014.