Deep Dictionary Learning

by
Snigdha Tariyal

Under the Supervision of Dr. Angshul Majumdar

Indraprastha Institute of Information Technology, Delhi
July, 2016

Deep Dictionary Learning

By
Snigdha Tariyal

Submitted

in partial fulfillment of the requirements for the degree of
Master of Technology

to

Indraprastha Institute of Information Technology Delhi

July, 2016

Certificate

This is to certify that the thesis titled " Deep Dictionary Learning " being submitted by Snigdha Tariyal to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

July, 2016

Dr. Angshul Majumdar
Department of Electronics and Communication
Indraprastha Institute of Information Technology Delhi
New Delhi 110 020

Contents

Certificate i
Acknowledgements.....iii
Abstract.....iv
List of Figures.....v
List of Tables.....vi

1. Introduction.....1
1.1. Thesis Roadmap.....3

2. Dictionary Learning and Deep Learning – an Overview.....5
2.1 Dictionary Learning.....5
2.1.1. K-SVD.....8
2.1.2. Sparse Representation based Classification (SRC).....10
2.1.3. Label Consistent KSVD (LCKSVD).....11
2.2 Deep Learning.....13
2.2.1 Motivation.....13
2.2.2 Deep Architectures.....14
2.2.2.1 Deep Boltzmann Machines.....14
2.2.2.2 Stacked Autoencoders.....17

3. Deep Dictionary Learning.....20
3.1 Overview.....20
3.2 Initialization of Dictionary.....23
3.3 Activation Function.....25
3.4 Algorithm 1 : Deep Dictionary Learning.....26
3.4.1 Algorithm for Training and Testing.....28
3.5 Algorithm 2 : Robust Deep Dictionary Learning.....29
3.5.1 Split-Bregman Algorithm.....30

3.5.2 Algorithm for Training and Testing.....31
3.6 Going Deep in Dictionary Learning.....33
4. Deep Dictionary Learning on Images.....35
4.1 Data Description.....35
4.1.1 MNIST.....35
4.1.1.1 Experimental Results on MNIST and its variations.....37
4.1.2 USPS.....40
4.1.2.1 Experimental Results 40
4.2 Effect of Deep Learning.....43
5. Deep Dictionary Learning on Hyperspectral Images.....45
5.1 Hyperspectral Data.....46
5.2 Data Description.....47
5.2.1 Indian Pines.....47
5.2.2 Pavia University.....49
5.3 Input Features.....50
5.4 Experimental Results.....52
5.5 Effect of Deep Learning.....59

Bibliography.....63

Acknowledgement

I would like to thank many people who helped me along my path to writing this thesis.

I would like to express my deepest gratitude to my thesis advisor , Dr Angshul Majumdar, for taking me under his wings, encouraging me and running a lab where all the students are free to explore creative ideas.

I sincerely thank all my professors in IIITD for the courses they taught, and to IIITD ,in general, for providing best facilities , quality education and a fun and intellectual atmosphere.

Several members of the SALSA Lab provided me with invaluable suggestions time to time. I would especially like to thank Protim Bhattacharjee , Hemant Aggarwal and Anupriya Gogna for their insightful comments and critical reviews.

A special thanks to all my friends for their patience and support when I doubted myself.

Above all, I would like to thank my family for being the pillar of my life. I thank my parents and brother for guiding me through life, filling me with positive energy and encouraging me to pursue my dreams.

Abstract

This Thesis focuses on combining the two well researched concepts of representation learning – Dictionary Learning and Deep Learning. These two learning paradigms have been known for long. Ever since, plethora of papers have been published for both the paradigms for solving inverse problems and for prediction problems. While dictionary learning focuses on learning "basis" and "features" by matrix factorization, deep learning focuses on extracting features via learning "weights" or "filter" in a greedy layer by layer fashion. While dictionary learning is shallow learning, deep learning methods are data hungry. This Thesis merges these two learning methodologies and proposes a new learning method referred to as Deep Dictionary Learning which tries to eliminate the disadvantages of the two. The proposed learning method learns multiple levels of representations using dictionary learning that correspond to different levels of abstraction; the levels form a hierarchy of linear/non-linear transformations.

On the above described strategy, two deep dictionary learning methods have been proposed in this thesis. Algorithm 1 is the deep dictionary learning method while Algorithm 2 is the robust version of algorithm 1, referred to as Robust Deep Dictionary Learning. Algorithm 2 consists of an additional denoising layer at the top most level for the purpose of generating useful representations even with noisy data.

The proposed techniques are compared with other learning approaches such as Stacked Auto Encoder, Deep Belief Network, LCKSVD1 and LCKSVD2 on benchmark datasets in the presence and absence of impulse noise of varying amounts. The classification performance of the proposed technique achieves higher or at par accuracies for both the cases. On a real world problem of hyperspectral image classification, it is observed that the proposed deep dictionary learning methods performs at par with other learning techniques with much less complexity and time in both, the absence and the presence of shot noise.

Also, the effect of going deep in dictionary learning versus shallow learning is discussed and shown experimentally.

List of Figures

Figure 1 : Synthesis Dictionary Learning.....9

Figure 2: Restricted Boltzman Machine.....15

Figure 3 : Deep Boltzman Machine.....16

Figure 4: Single layer Autoencoder.....17

Figure 5 : Stacked Autoencoder.....18

Figure 6 : Schematic Diagram for Dictionary Learning.....21

Figure 7: Schematic Diagram for Deep Dictionary Learning for two layers..21

Figure 8: Greedy Layer-wise Dictionary Learning.....23

Figure 9 : sigmoid function & tanh function 25

Figure 10 : MNIST dataset.....35

Figure 11 : USPS dataset.....40

Figure 12: Hyperspectral Datacube.....47

Figure 13 : a) False color image of Indian Pines (band 50,27,17).....49
b) Ground truth representing 16 crop cases

Figure 14: a) False-color image of Pavia, Italy (Band 10, 27, 46) 49
b) Ground truth representing 9 land-cover classes.

Figure 15: Spectral vector of Hyperspectral Features.....50

Figure 16: Spatial Vector of Hyperspectral Features.....51

Figure 17: Spectral Spatial Vector of Hyperspectral Features.....52

Figure 18,19,20,21: Graph for Classification Accuracy for Case 2(A).....61

Figure 22,23,24,25: Graph for Classification Accuracy for Case 2(B).....62

List of Tables

Table 3.1: Training Algorithm for Deep Dictionary Learning.....28

Table 3.2: Testing Algorithm for Deep Dictionary Learning.....29

Table 3.3: Training Algorithm for Robust Deep Dictionary Learning.....31

Table 3.4: Testing Algorithm for Robust Deep Dictionary Learning.....32

Table 4.1: Classification accuracy on MNIST and its variants
for proposed methods.....37

Table 4.2: Classification accuracy on MNIST and its variants
for proposed methods with added impulse noise of 10% to
training and testing sets.....38

Table 4.3: Classification accuracy on MNIST and its variants
for proposed methods with added impulse noise of 20%
to both training and testing sets.....39
.
Table 4.4: Classification accuracy on MNIST and its variants
for proposed methods with added impulse noise of 30%
to both training and testing sets.....39

Table 4.5: Classification accuracy on USPS for proposed methods
with SVM as classifier.....40

Table 4.6: Classification accuracy with added 50% impulse noise
to training samples and testing samples for proposed methods
using KNN as classifier.....41

Table 4.7: Classification accuracy with added 60% impulse noise
to training samples and testing samples for proposed methods using KNN
classifier..... 42

Table 4.8: Classification accuracy with added 70% impulse noise
to training samples and testing samples for proposed methods
using KNN as classifier..... 42

Table 4.9: Classification accuracy with added 50% impulse noise
to testing samples for proposed methods using KNN as classifier..... 42

Table 4.10: Classification accuracy with added 60% impulse noise

to testing samples for proposed methods using KNN as classifier..... 43

Table 4.11: Classification accuracy with added 70% impulse noise
to testing samples for proposed methods using KNN as classifier..... 42

Table 4.12: Classification accuracy on USPS and MNIST and
its variations for Shallow vs Deep learning for proposed methods.....44

Table 5.1: Groundtruth classes of Indian Pines along with
the sample number of each class.....48

Table 5.2: Groundtruth classes of Pavia University along
with the sample number of each class.....50

Table 5.3: Classification accuracy with 10% training samples and
90% testing samples for proposed methods using KNN as classifier.....53

Table 5.4: Classification accuracy with 20% training samples and
80% testing samples for proposed methods using KNN as classifier.....54

Table 5.5: Classification accuracy with added noise to 10% training
samples and 90% testing samples for proposed methods using KNN
as classifier.....55

Table 5.6: Classification accuracy with added noise
to 20% training samples and 80% testing samples for
proposed methods using KNN as classifier.....55

Table 5.7: Classification accuracy with added noise to
10% training samples and 90% testing samples for
proposed methods using KNN as classifier.....56

Table 5.8: Classification accuracy with added noise to
20% training samples and 80% testing samples for
proposed methods using KNN as classifier.....56

Table 5.9: Classification accuracy with added noise to
10% training samples and 90% testing samples for
proposed methods using KNN as classifier.....56

Table 5.10: Classification accuracy with added noise to
20% training samples and 80% testing samples for
proposed methods using KNN as classifier.....57

Table 5.11: Classification accuracy with 10% training samples
and added noise to 90% testing samples for proposed methods

using KNN as classifier.....57

Table 5.12: Classification accuracy with 20% training samples and added noise to 80% testing samples for proposed methods using KNN as classifier.....57

Table 5.13: Classification accuracy with 10% training samples and added noise to 90% testing samples for proposed methods using KNN as classifier.....58

Table 5.14: Classification accuracy with 20% training samples and added noise to 80% testing samples for proposed methods using KNN as classifier.....58

Table 5.15: Classification accuracy with 10% training samples and added noise to 90% testing samples for proposed methods using KNN as classifier.....58

Table 5.16: Classification accuracy with 20% training samples and added noise to 80% testing samples for proposed methods using KNN as classifier.....59

Table 5.17: Classification accuracy on Indian Pines for Shallow vs Deep learning for proposed methods.....59

Table 5.18: Classification accuracy on Pavia University for Shallow vs Deep learning for proposed methods.....60

1 | Introduction

With the dramatic increase in the size of the data, the resourcefulness of such information is dependent on how well any knowledge can be extracted from it. We are moving towards the idea of extracting and organizing discriminative information from the data. Recently, there has been growing interest in learning models based on domain knowledge and applications. The performance of any learning algorithm nowadays depends heavily on the features or data representation on which they are applied on. For that reason, much of the actual effort in learning algorithms goes into the design of preprocessing pipelines and data transformations that result in a representation of the data that can support effective learning. This thesis is about representation learning, i.e., learning representations of the data that make it easier to extract useful information when building classifiers or other predictors[7]. Representation learning algorithms have been applied to many area such as Speech recognition, object recognition, natural language processing etc.

In representation learning paradigm, dictionary learning has generated a lot of interest. The concept of dictionary learning has been around for much longer when the researchers were using the term 'matrix factorization'. The goal was to learn an empirical basis from the data. It basically required decomposing the data matrix to a basis / dictionary matrix and a feature matrix; hence the name 'matrix factorization'. In recent years, the idea of the adaptivity has been exploited to design the dictionary specifically optimized for the target dataset. This was the alternative from the reliant, tried and tested off the shelf dictionaries .The data adaptive approach claims to be more efficient in facilitating better learning thus leading to better interpretation of the data [7]. A

reasonably-sized learned representation can capture a huge number of possible input configurations.

Dictionary learning can be used both for unsupervised problems (mainly inverse problems in image processing) as well as for problems arising in supervised feature extraction[6].

Prior studies on dictionary learning (DL) are, generally, 'shallow or surface' learning models just like a restricted Boltzmann machine (RBM) [4] and Autoencoder (AE) [5]. In DL, the cost function is Euclidean distance between the data and the representation given by the learned basis and a sparse representation; for RBM it is Boltzmann energy; whereas for AE, the cost is the Euclidean reconstruction error between the input data and the decoded representation/features [6].

Almost at the same time, one new parameter in learning paradigm gained popularity. The parameter was 'Depth' . It paved way for another representation learning paradigm- Deep Learning. Deep Learning means constructing multiple levels of representation or learning a hierarchy of features. The depth of the architecture is the length of the longest path from an input node to an output node[7]. Deep Belief Network (DBN) is formed by stacking one RBM after the other [8,9]. Similarly, Stacked Autoencoder (SAE) are created by one AE followed by the other [10,11].

The advantages of Deep learning are [7] :

(1) promotion of re-use of features,

(2) a sufficiently deep architecture can potentially lead to more abstract features at higher layers of representations .

Deep architectures are often challenging to train effectively and this has been the subject of much recent research and progress [7].

Inspired from both the feature learning strategies, in this thesis, we propose to learn multi-level deep dictionaries. Our objective here is to learn the better representation of the data which then is used in forming deep architectures.

The learning of deep representations can be divided into two aspects: unsupervised feature learning and supervised deep learning. The thesis proposes the unsupervised deep learning which generates features at the output. A hierarchy of features is learnt one level at a time, using unsupervised dictionary learning to learn a new linear/non-linear transformation at each level composed with the previously learned representations. Finally, the representations from the last layer can be given as input to any classifier for the task of classification or simply used for other purposes such as denoising and likes.

It is many times argued that multiple levels of dictionaries can be collapsed to a single level dictionary. However such a collapsed shallow dictionary will not be the same as the proposed deep dictionary learning. This is because dictionary learning is a bi-linear problem. Had it been linear the architecture would have been collapsible; since it is not, the shallow and the deep architectures will not be equivalent [6]. There are few results in the Thesis that will prove so.

1.1 Thesis Roadmap

Chapter 2 - Dictionary Learning and Deep Learning –an Overview

The motivation and overview of the existing approaches for Dictionary Learning and Deep Learning.

Chapter 3 – Deep Dictionary Learning

The proposed deep Dictionary Learning Algorithms inspired from dictionary learning and deep learning strategy are explained and explored by mathematical formulations

Chapter 4 – Deep Dictionary learning on Images

Experimental evaluation of benchmark dataset MNIST and handwritten dataset USPS for classification and few results for denoising. The results are compared with State-of-the-art techniques such as SAE and DBN.

Chapter 5 – Deep Dictionary learning on Hyperspectral Images

Experimental evaluation of Hyperspectral dataset Indian Pines and Pavia University for classification and few results for denoising with realistic scenario of ratio of training and testing being 10:90 and 20:80 .The results are compared with State-of-the-art techniques such as SAE and DBN.

2 | Dictionary Learning and Deep Learning – an Overview

The surging interest in various 'Learning' techniques due to explosive growth in volumes and varieties of available data, cheaper and more powerful computational processing , and affordable data storage has led to the popularity of Dictionary Learning. The technique is especially evolving for numerous low-level tasks such as denoising [12], texture synthesis [13], and audio processing [14] as well as higher-level tasks such as Classification [15]. Recent Dictionary Learning techniques allow flexibility of basis vectors to adapt the representation to the data rather than fixed basis. Advancement in learning procedures has led to unsupervised as well as supervised Dictionary Learning algorithms.

2.1 Dictionary Learning

Dictionary Learning is essentially the technique to learn the (often linear) combination of basis elements, the so-called atoms, to adapt it to specific data. The collection of these atoms is called a dictionary .Early studies in Dictionary learning focused on learning only basis for representation. Usually the dictionary is initialized using randomly selected elements from data itself.
The approximation of the data can be written as :

$$X \approx DZ \qquad \ldots (1)$$

where $X = [x_1 , x_2 , x_3 , \ldots\ldots x_n]$, $x_i \in R^m$, set of data vectors

$Z = [z_1 , z_2 , z_3 \ldots\ldots\ldots z_n] \in R^{pXn}$, representation coefficients

$D = [d_1 , d_2 , d_3 \ldots\ldots\ldots d_p] \in R^{mXp}$, set of basis vectors called Dictionary

The learning problem was formulated as unconstrained Euclidean cost function measuring the quality of representation of the data. The Method of Optimal Directions [16] was used to learn the Dictionary :

$$\min_{D,Z} \|X - DZ\|_F^2 \qquad \ldots (2)$$

where $\|*\| = \sum_{i=1}^{m}\sum_{j=1}^{n}|*_{i,j}|$, also called the Frobenius Norm of the Matrix.

Equation (2) is easily solved using alternating minimization [20]. In every iteration, the first step is to update the coefficients through least squares assuming D is fixed and the next step is to update the dictionary through least squares assuming the coefficients are fixed. This alternating update of dictionary and coefficients continues till the algorithm converges to some local minima.

Before the data driven dictionaries were widely accepted , the general practice was to use off-the shelf fixed dictionaries, e.g. local Discrete Cosine Transform (DCT) [1], Wavelets [2,3] , Curvelets [17,18], and Wedgelets [19]. The advantage of such basis from a computational point of view is that they compute the representation coefficients by calculating a scalar product between the input data and the dictionary atoms i.e faster transforms, though their main disadvantage is limiting the choice of atoms and thus sometimes not enough representation of the data.

Equation (2) is also known as Matrix Factorization. There are no constraints on the loading coefficients. The dictionary, D defined above can be "undercomplete" if m > p or "overcomplete" if m < p. With the Overcomplete dictionaries the condition for the atoms to be orthogonal is relaxed thus allowing for more flexible dictionaries and richer data representations.

More recent works have focused on representing data vectors as linear combinations of few elements from *dictionary*, introducing the concept of *sparsity* (solutions having a minimum number of nonzero elements) [21,22].The focus is now shifted on how to sparsely encode a signal given the dictionary i.e. Z needs to be sparse. If the dictionary is large and rich enough in representational power, data can be matched to a very few (perhaps even just one) dictionary atom(s).

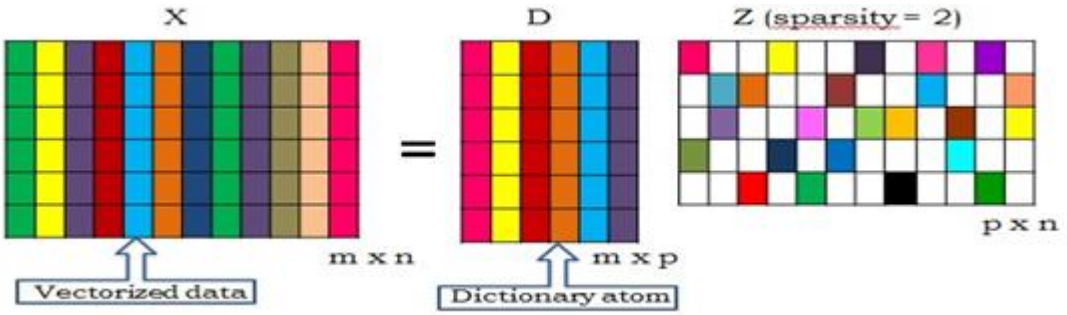


Figure 2: Synthesis Dictionary Learning

The framework can be written as :

$$\min_{D,Z} \|X - DZ\|_F^2 \quad s.t \quad \|Z\|_0 \leq \tau \qquad \ldots.(3)$$

Here Z is τ sparse and $\|Z\|_0$ is the l_0- norm of a vector which counts the no. of non-zero elements in it. l_0- minimization is regarded by computer scientist as an NP-hard problem [24], simply says that it's too complex and almost impossible to solve. Greedy techniques are available for solving such problems. Popular among them are Orthogonal Matching Pursuit (OMP) [23] and its variants [25-27]. However, In many cases, l_0 -minimisation problem is relaxed to be a higher-order norm problem such as l_1 – minimization. For a 'n' dimensional vector X ,the l_1 – norm is defined as $\sum_{i=1}^{n} |x_i|$.

Now the problem becomes convex and thus the optimization is of the following form :

$$\min_{D,Z} \|X - DZ\|_F^2 \quad s.t \quad \|Z\|_1 \leq \tau \qquad \ldots.(4)$$

We solve Equation (4) by relaxing the constraint via the Lagrangian method and thus can be re-written as:

$$\min_{D,Z} \|X - DZ\|_F^2 + \lambda \|Z\|_1 \quad \text{....(5)}$$

Where λ is the Langrangian multiplier. We then solve for sparse representation (Z) and Dictionary (D) iteratively. Given D, estimating Z reduces to the sparse coding problem. For sparse coding, Equation (5) is known as LASSO (Least Angle Shrinkage and Selection Operator) [28]. The literature is vast for l_1 minimization techniques like soft-thresholding based methods [29,30], reweighted-l_2 methods [31] etc. Given Z, estimating D is a least squares problem.

There has been significant interest in finding sparse solutions to the signal representation problem.Experiments have shown that such a model with sparse decompositions (*sparse coding*) is very effective in many applications. Dictionary Learning and Sparse coding are used interchangeably now.

Over the years many Algorithms for data-driven learning of dictionaries have been proposed. Following is a brief review of a few of them:

2.1.1 K-SVD

One of the pioneering work in Dictionary Learning is the K-SVD method [22]. The Algorithm learns an over-complete dictionary as well as the sparse representations of the patches under that dictionary in an alternating minimization framework.

Fundamentally, it solves a problem of the form:

$$\min_{D,Z} \|X - DZ\|_F^2 \quad s.t \quad \|Z\|_1 \leq \tau \quad \text{...(6)}$$

In Equation (6), X is the matrix of vectored training data and D is the learnt dictionary and matrix Z represents the coefficients and column sparse. If n is the number of training samples, then X has the dimension of mxn, dictionary has a dimension of m x p, with m << p and Z has a dimension of m x n.

Equation (6) is solved in two steps. The update of the dictionary columns is combined with an update of the sparse representations, thereby accelerating convergence[22]. The K-SVD algorithm is flexible and can work with any pursuit method (e.g., basis pursuit, FOCUSS, or other matching pursuits)[31-34].

In the first stage it learns the dictionary and in the next stage it uses the learned dictionary to sparsely represent the data. Solving the l_0-norm minimization problem is NP hard [24]. K-SVD employs the greedy (sub-optimal) orthogonal matching pursuit (OMP) [23] to solve the l_0 -norm minimization problem approximately with predetermined sparsity. In the dictionary learning stage or codebook update stage, K-SVD proposes an efficient technique to estimate the atoms one at a time using a rank one SVD update with sparsity constraints[22]. Such dictionary learning methods have achieved state of the art performances both synthetic and real images in applications such as filling in missing pixels and compression and outperforms alternatives such as the Stationary wavelet transform and overcomplete or unitary DCT. [35-38]

The major disadvantage of K-SVD is that it is a relatively slow technique owing to its requirement computing the SVD (singular value decomposition) in every iteration.

Over the years Dictionary learning has been extensively used in the field of Image denoising[38] , Additive noise removal (low light), Multiplicative noise removal, Video denoising , Image restoration , Image inpainting [39] , image half-toning , Block Artifact removal and likes. The expanding learning scenario introduced Dictionary learning to Classification tasks as well. The Dictionary learning for classification can be supervised , unsupervised or semi-supervised.

The dictionary learning formulation in equation (6) is unsupervised. There is a large volume of work on supervised dictionary learning problems like Sparse Representation Classifier (SRC), Discriminative KSVD (DKSVD) and Label Consistent KSVD (LC-KSVD).

2.1.2 Sparse Representation based Classification (SRC)

Sparse Representation based Classification (SRC) [40] is not much of a "dictionary learning technique", but a simple dictionary design problem where all the training samples are concatenated in a large dictionary.
Among all the atoms in an over complete dictionary, the sparse representation selects the subset of the atoms which most compactly expresses the input signal and rejects all other less compact representation. Therefore, the sparsest representation of a signal is naturally discriminative and can be developed for signal classification purpose. Sparse representation classifier is a nonparametric learning method which can directly predict or assign a class label to a test sample based on dictionary composed of training samples[6].

In sparse representation classifier, the dictionary is constructed from training samples from various classes. The j^{th} class training samples are arranged as column of a matrix D_j as shown :

$$D_j = \left[d_{j,1}\, d_{j,2} \ldots\ldots\ldots d_{j,n} \right] \quad \in R^{mxn_j} \qquad \ldots(7)$$

where $d_{j,i}$ denotes the training sample belonging to the jth class and nj is number of training samples belonging to jth class. The dictionary, D is form using all the dictionary from each class as shown:

$$D = \left[D_1\, D_2 \ldots\ldots\ldots D_c \right] \quad \in R^{mxn} \qquad \ldots(8)$$

Where $n = \sum_{j=1}^{c} n_j$ and c is the number of classes.

The entries of Z that corresponds to the class which the test sample D belongs to is expected to be nonzero while the entries of that corresponding to other classes is expected to be zero. The minimization problem is then casted as :

$$\min_{D,Z} \|X - DZ\|_F^2 \quad s.t \quad \|Z\|_1 \leq \tau \qquad \ldots(9)$$

And solved by relaxing the constraint by Langrangian multiplier method :

$$\min_{D,Z} \|X - DZ\|_F^2 + \lambda \|Z\|_1 \qquad \ldots(10)$$

Equation 10 is a LASSO problem and can be solved using spectral projected gradient method, SPGL1 toolbox or other available methods. The minimum of the representation error or the residual error of class c is calculated by keeping the coefficients associated with that class and while setting the other entries to zero. This is done by introducing a characteristic function, ζ as :

$$r_c(x) = \|x - D_j \zeta_j(z)\|_2 \qquad \ldots(11)$$

where $r_c(x)$ denotes the residual error. The vector ζ has value one at locations associated to the class j and zero for other entries. The class, d, of the test signal, D is computed as the one that produces smallest residual error.

$$d = \min_j r_j(x)$$

Sparse representation for classification was first introduced in 2009 in face recognition research and since then been quite popular.

2.1.3 Label Consistent KSVD (LC-KSVD)

The label consistent KSVD is one of the more recent techniques for learning discriminative sparse representation. It is simple to understand and implement; it showed good results for face recognition [41,42].

The first technique is termed as Discriminative K-SVD [41] or LC-KSVD1 [42]; it proposes an optimization problem of the following form:

$$\min_{D,Z,A} \|X - DZ\|_F^2 + \lambda_1 \|D\|_F^2 + \lambda_2 \|Z\|_1 + \lambda_3 \|Q - AZ\|_F^2 \qquad \ldots(12)$$

Here, Q is the label of the training samples; it is a canonical basis with a one for the correct class and zeroes elsewhere. A is a parameter of the linear classifier. It forces the signals from the same class to have very similar sparse representations (i.e., encouraging label consistency in the resulting sparse codes), which results in good classification performance even using a simple linear classifier.

A second formulation is proposed that adds another term to penalize the classification error. Classification error is introduced as a term in the cost function for dictionary learning to make the dictionary optimal for classification.The LC-KSVD2 [42] formulation is as follows:

$$\min_{D,Z,A} \|X - DZ\|_F^2 + \lambda_1 \|D\|_F^2 + \lambda_2 \|Z\|_1 + \lambda_3 \|Q - AZ\|_F^2 + \lambda_4 \|H - WZ\|_F^2 \qquad \ldots(13)$$

Here H_i is a 'discriminative' sparse code corresponding to an input signal sample, if the nonzero values of H_i occur at those indices where the training sample X_i and the dictionary item d_k share the same label. W denotes the classifier parameters. Basically this formulation imposes labels not only on the sparse coefficient vectors Z_i's but also on the dictionary atoms[6].

The dictionary learned in this way is adaptive to the underlying structure of the training data (leading to a good representation for each member in the set with strict sparsity constraints), and generates discriminative sparse codes Z and addresses the desirable property of the discriminability of classifier construction regardless of the size of the dictionary. These sparse codes can be utilized directly by a classifier. The discriminative property of sparse code Z is very important for the performance of a linear classifier [6].

Prior studies on dictionary learning (DL) are, generally, 'shallow' or 'surface' learning models . Almost at the same time, when dictionary learning started gaining popularity, researchers in machine learning observed that better (more abstract and compact) representation can be achieved by going deeper in a

neural network architecture. Deep Belief Network (DBN) is formed by stacking one RBM after the other [8,9]. Similarly, stacked autoencoder (SAE) are created by one AE followed by the other [10,11].

2.2 Deep Learning

Deep learning is a recently popular research topic in the machine learning research community. The successful applications in the direction of many fields such as Automatic Speech recognition, Image recognition, Bioinformatics, Natural language processing, Recommender system are all to do with the advantage of deep learning acting as a good feature representation method.

2.2.1 Motivation

Most of the conventional data representations are all in the form of handcrafted features, which are too heuristic to be adaptive to the various input data. Many feature learning methods try to use human prior knowledge to improve the performance. However, these features have limited performance in terms of obtaining discriminative information from the input data. In deep architectures, according to the target in the output layer, this deep learning process can help to learn a more representative model. For instance, the model could have edge detectors in the first layer, more abstract feature in the second layer, and then succeeding layers with more abstract features. Thus high-level abstractions in data is modeled by using a deep architecture with multiple processing layers, composed of multiple linear and non-linear transformations. Owing to the commercial profits foreseen, technology giants, such as Google, Chinese search company, Baidu ,Facebook and Apple, have started actively pursuing research in this area.

2.2.2 Deep Architectures

Deep Learning uses a cascade of many layers of linear/nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input. The algorithms may be supervised or unsupervised .The foundation is based on the (unsupervised) learning of multiple levels of features or representations of the data. Higher level features are derived from lower level features to form a hierarchical representation.It learns multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts. It also appears sensible to learn simple representations first and higher-level abstractions on top of existing lower level ones. In place of randomly initialized parameters, this representation forms the initialization – a catalyst to learn meaningful representations – for the subsequent learning phase[6].

2.2.2.1 Deep Boltzmann Machines

Restricted Boltzmann Machines [8,9] are undirected models that use stochastic hidden units to model the distribution over the stochastic visible units. The hidden layer is symmetrically connected with the visible unit and the architecture is restricted" as there are no connections between units of the same layer. Traditionally, RBMs are used to model the distribution of the input data p(x).

Consider a binary RBM with a visible input layer X and a latent/hidden layer H. The input layer contains I dimensions corresponding to the size of the input vector. The latent layer has J latent variables. Additionally, there are offset (or bias) units, x_0 and z_0, that are permanently set to one. The layers are associated by an undirected weight matrix W, such that every input unit i is connected to every latent variable j via w_{ij}.

The schematic diagram of RBM is shown in Fig. 2

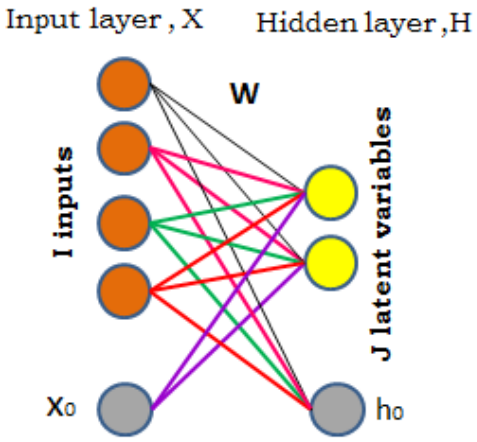


Figure 2: Restricted Boltzman Machine

The objective is to learn the network weights (W) and the representation (H). This is achieved by optimizing the Boltzmann cost function given by:

$$p(W,H) = e^{-E(W,H)} \qquad \ldots(14)$$

Where $E(W,H) = -H^T WX$ including the bias terms. Assuming independence, the conditional distributions are given by :

$$p(X|H) = \prod p(x|h)$$

$$p(H|X) = \prod p(h|x)$$

Given an input vector, the activation probabilities of the latent units can be sampled:

$$p(h_i = 1|x;W) = \frac{1}{1+\exp(-\sum_{i=0}^{I} w_{ij} x_i)} = sigm(Wx)$$

While the input units can be sampled from the latent/hidden vector with a symmetric decoder:

$$p(x_i = 1|h;W) = \frac{1}{1+\exp(-\sum_{j=0}^{J} w_{ij} h_j)} = sigm(W^T h)$$

Computing the exact gradient of this loss function is almost intractable. However, there is a stochastic approximation to approximate the gradient termed as contrastive divergence gradient. A sequence of Gibbs sampling based reconstruction, produces an approximation of the expectation of joint energy distribution, using which the gradient can be computed.

Usually RBM is unsupervised, but there are studies where discriminative RBMs are trained by utilizing the class labels [43]. There are also RBMs which are sparse [44]; the sparsity is controlled by firing the hidden units only if they are over some threshold. Supervision can also be achieved using sparse RBMs by extending it to have similar sparsity structure within the group / class [45].

Deep Boltzmann Machines (DBM) [46] is an extension of RBM created by stacking multiple hidden layers on top of each other (Fig. 2). DBM is an undirected learning model and thus it is different from the other stacked network architectures in which each layer receives feedback from both the top-down and bottom-up layer signals. This feedback mechanism helps in managing uncertainty in learning models.

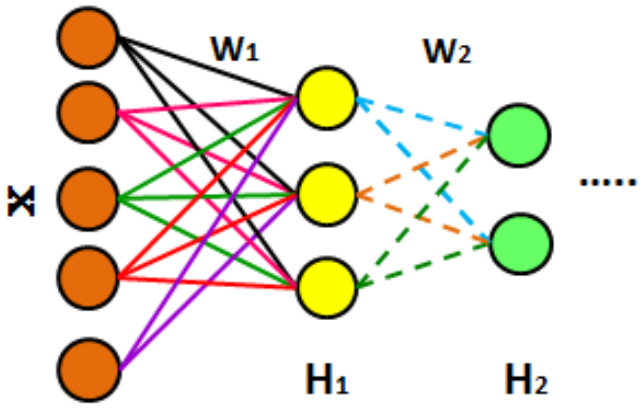


Figure 3 : Deep Boltzmann Machine

2.2.2.2 Stacked AutoEncoder (SAE)

The quintessential example of a representation learning algorithm is the Autoencoder[10,11]. The network is trained to reconstruct its inputs, which forces the hidden layer to try to learn good representations of the inputs. They are trained to reconstruct their own inputs. An autoencoder consists of three layers: i) an input layer , ii) an encoding layer (hidden layer) that maps the input data into a latent representation, and iii) a decoding layer that maps the learnt representation back into the original data.

For a given input vector (including the bias term) x, the latent code or variable is expressed as:

$$h = Wx$$

Here, the rows of W are the link weights from all the input nodes to the corresponding latent node. Usually the mapping function(Φ) is non-linear at the output of the hidden nodes leading to:

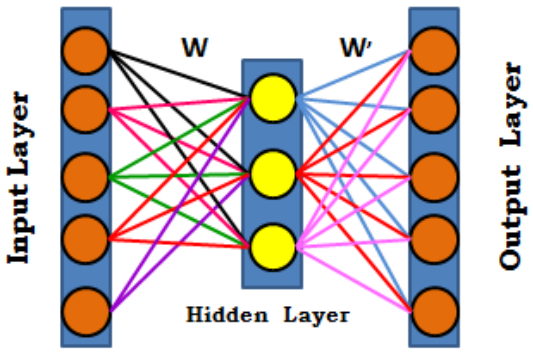


Figure 4: Single layer Autoencoder

$$h = \Phi(W x + b) \qquad \ldots(15)$$

W is the weight matrix and b is the offset vector. Although a non-linear function is popularly used, a linear activation functions is also used sometimes depending on input x [10].

The decoder reverse maps the latent variables to the data space.

$$x = W' h + b'$$

Or, $$x = W' \Phi(W x + b) + b' \quad \ldots(16)$$

During training, the problem is to learn the encoding and decoding weights – W and W'. These are learnt by minimizing the Euclidean cost:

$$\min_{W,W'} \left\| X - (W' \Phi(W X + b) + b') \right\|_F^2 \quad \ldots(17)$$

The problem in Equation (17) is clearly non-convex, but as the activation function is generally smooth and continuously differentiable, it can be solved by gradient descent techniques.

The idea of autoencoders was extended to deep and thus Stacked autoencoders were introduced. Stacked/Deep autoencoders have multiple hidden layers – one inside the other (see Fig. 4). The corresponding cost function is expressed as follows in Equation 18:

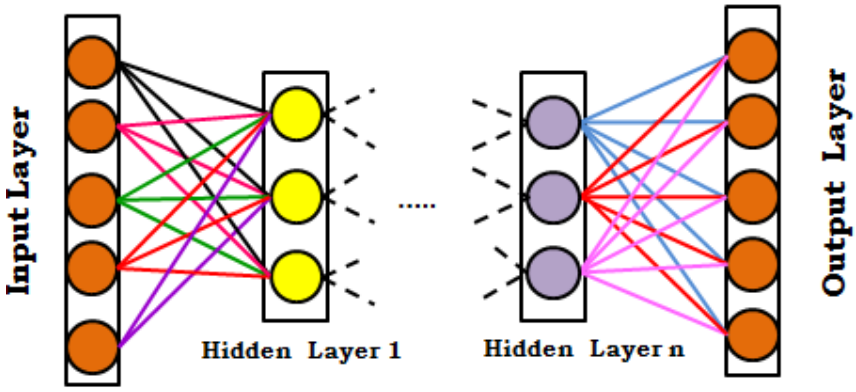


Figure 5 : Stacked Autoencoder

$$\min_{W_1 .. W_n, W_1' ... W_n'} \left\| X - g \circ f(X) \right\|_F^2 \quad \ldots(18)$$

Where $$g = W_1' \Phi(W_2' .... W_n'(f(X)) \quad and$$
$$f = \Phi(W_n \Phi(W_{n-1} ..... \Phi(W_1 X))) \quad \ldots(19)$$

Solving the complete problem (17) is computationally challenging. Also learning so many parameters (network weights) lead to over-fitting. To address both these issues, the weights are usually learned in a greedy layer by layer fashion. The process of finding these initial weights is often referred to as pre-training. After finding the deviation of input from the output, error is backpropagated through the network and weights are updated. As errors are backpropagated to through the layers, they are minimized. Thus the network almost always learns to reconstruct the average of all the training data.

There are several variations to the basic autoencoder architecture like Denoising Autoencoder[11], Sparse Autoencoder[61,62] and Variational Autoencoder.

Stacked denoising autoencoder [11] is a variant of the basic autoencoder where the input consists of noisy samples and the encoder and decoder are learnt to reconstruct the original input.

Another variation for the basic autoencoder is to regularize it, i.e.

$$\min_{W_1..W_n, W_1'...W_n'} \left\| X - g \circ f(X) \right\|_F^2 + R(W, X) \qquad \ldots(20)$$

The regularization in Equation (20) can be a sparsity promoting term [61, 62]. The regularization term is usually chosen so that they are differentiable and hence minimizable using gradient descent techniques.
Different kinds of autoencoders aim to achieve different kinds of properties.

3 | Deep Dictionary Learning

In this Chapter, we describe the main contribution of this Thesis. A single / shallow level of dictionary learning yields a latent representation of data and the dictionary atoms. Here, it is proposed to learn latent representation of data by learning multi-level dictionaries[6]. The idea of learning deeper levels of dictionaries stems from the success of deep learning. Based on this, we build a deep architecture by cascading one dictionary after the other. The learning proceeds in a greedy fashion, therefore for each level only a single layer of dictionary is needed to learn. There are time tested tools to solve this problem.

Both deep learning and dictionary learning fall under the broader category of representation learning. Representation learning means automated feature extraction. Classical feature extraction techniques were either based on statistical models (Principal Component Analysis[47], Linear Discriminant Analysis[48] etc.) or were hand-crafted (Scale Invariant Feature Transform(SIFT) [49], Local Binary Pattern[50] etc.). Classical feature extraction techniques were 'designed' in the sense that it is based on some assumption / model made by the researcher / engineer regarding the nature of the data. On the other hand representation learning, 'learns' the model by itself, given the training data. As Deep Dictionary learning is the combination of deep learning and dictionary learning, it also falls in the category of Representation Learning.

3.1 Overview

In this section, for ease of understanding, the concept with two-layer deep dictionary learning is explained and then extended to multi-level dictionary.
The schematic diagram for dictionary learning is shown in Fig.5 . Let X be the set of data vectors, D be the data-adaptive dictionary and Z be the

feature/representation of X in D. Dictionary learning follows a synthesis framework (Equation 21), i.e. the dictionary is learnt such that the features synthesize the data along with the dictionary.

$$X = DZ \qquad \ldots(21)$$

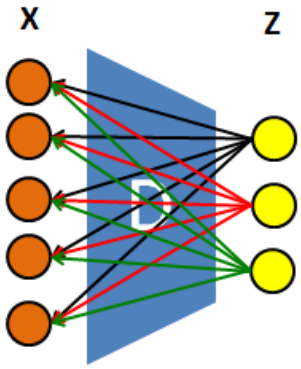


Figure 6 : Schematic Diagram for Dictionary Learning

The thesis proposes to extend the shallow dictionary (Fig. 5) learning into multiple layers – leading to deep dictionary learning (Fig. 6). For the first layer, a dictionary is learnt to represent the data. In the second layer, the representation from the first layer acts as input; it learns a second dictionary to represent the features from first level. This concept can be extended to deeper layers. Mathematically, the representation at the second layer can be written as:

$$X = D_1 D_2 Z_2 \qquad \ldots(22)$$

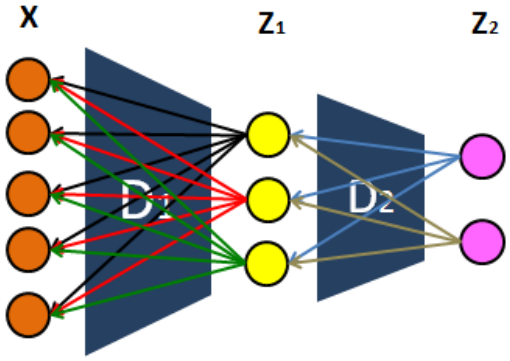


Figure 7: Schematic Diagram for Deep Dictionary Learning for two layers

It must be noted that learning two-levels of dictionaries along with the coefficients (Equation 22) is not the same as learning a single (collapsed) dictionary and its corresponding features. Problem (Equation 21) (single level) is a bi-linear problem and (Equation 22) is a tri-linear problem; they are not the same. Hence one cannot expect to get the same features from a single level dictionary learning and a collapsed two level dictionary learning.

The challenges of learning multiple levels of dictionaries in one go are the following:

1) Recent studies have proven convergence guarantees for single level dictionary learning [51-53]. These proofs would be very hard to replicate for multiple layers.

2) Moreover, the number of parameters required to be solved increases when multiple layers of dictionaries are learnt simultaneously. With limited training data, this could lead to over-fitting.

Here we propose to learn the dictionaries in a greedy manner which is in sync with other deep learning techniques [8-11]. Moreover, layer-wise learning will guarantee the convergence at each layer. The idea applies to deep in the sense that in the second layer, the representation from the first layer acts as input; it learns a second dictionary to represent the features from first level. Similarly representation from second layer acts as input to the third layer to learn third dictionary to find the features. This concept can be extended to deeper layers.

Extending this idea, a multi-level dictionary learning problem with non-linear activation (Φ) can be expressed as in Equation [23]

$$X = D_1\Phi\left(D_2\Phi(D_3\Phi.....\Phi(D_N Z))\right) \quad \ldots(23)$$

Here Φ can be a linear or non -linear activation function.

$$X = D_1\Phi\left(D_2\Phi(D_3\Phi.....\Phi(D_N Z))\right) \quad ...(23)$$

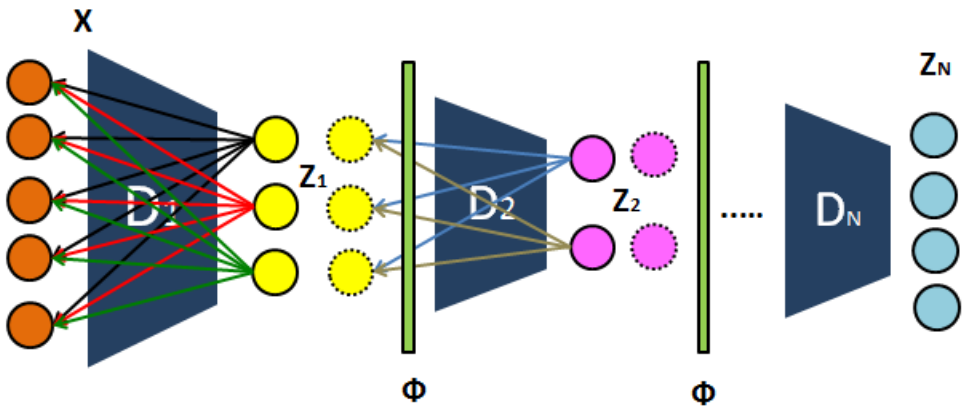


Figure 8: Greedy Layer-wise Dictionary Learning .

In deep learning, instead of learning the full deep architecture in one go, the parts are usually learnt in a greedy fashion [12]. There are two advantages of greedy learning.

1. Learning layer wise dictionary and features are relatively simple and easy to implement.
2. Deep architectures, having a large number of variables, are difficult to learn from limited training data. Breaking the problem into smaller unit erases the issue of over-fitting.

This Thesis has introduced two Algorithms merging two learning paradigms – Dictionary Learning and Deep Learning – as Deep Dictionary Learning - For primarily unsupervised Classification task.

3.2 Initialization of Dictionary

A variety of dictionaries have been developed in response to the rising need. These dictionaries emerge from one of two sources: either a mathematical model of the data, or a set of realizations of the data. Dictionaries of the first type are characterized by an analytic formulation and a fast implicit implementation, while dictionaries of the second type deliver increased flexibility and the ability to adapt to specific signal data. Most recently, there is a growing interest in dictionaries which can mediate between the two types,

and offer the advantages of both worlds. Such structures are just beginning to emerge, and research is still ongoing.

One of the interesting concept here is, as the idea of data driven dictionary is adapted for the algorithm, the dictionary D for any layer is initialized either from the input data (in the case of first layer) or from the features acting as input (in the case of successive layers). Depending on the size of dictionary , data vectors are chosen randomly to initialize the dictionary.

Another option is to initialize the dictionary from the samples of the Q matrix from the QR Decomposition of the input data. By input, it means the respective input to a layer. QR decomposition/factorization of a matrix ,is the factorization into a product, namely, Q x R where Q is an orthogonal matrix and R is an upper triangular matrix. QR decomposition is Gram–Schmidt orthogonalization of columns of A, starting from the first column. The property of Q is of great importance in the algorithm. If A has n linearly independent columns, the first k columns of Q form an orthonormal basis for the span of the first k columns of A for any $1 \leq k \leq n$ .Thus the initial dictionary is orthogonal to the input data and the representation coefficients can be computed as inner products of the signal and the atoms.

However, any one the initializations can be used to generate features and classify. Dictionaries of each layer can be initialized in either way. Dictionary initialization of each layer is independent of the dictionary initialization of its preceding or succeeding layer.

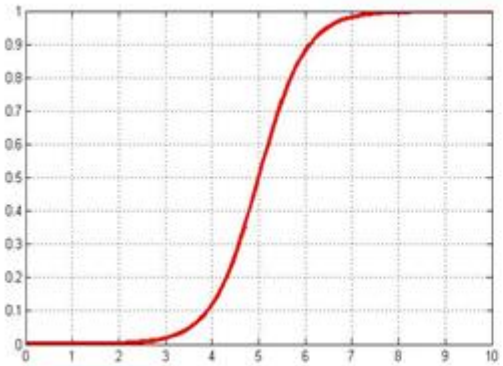
The discussion on dictionary initialization develops another compelling idea : whether to go for overcomplete (or fat) or undercomplete (or tall) dictionary. With the idea of initialization from the columns of Q matrix, the dictionary is, in majority of the cases , undercomplete as number of features are less than number of data vectors which are stacked as columns to make the input matrix. For example if: X = m x n , m < n then Q = m x m .To initialize the dictionary with columns of Q, the maximum number is ‘m’ with a square dictionary which makes no sense. Thus the only choice is to go for an

undercomplete dictionary. However if m > n, then either of the choice is open. Similarly, for dictionary initialization with the input data m < n , dictionary can be undercomplete or overcomplete. But with the reverse case, dictionary can only be undercomplete.

3.3 Activation Function (Φ)

It is essentially a function used to transform the input into an output signal. Activation functions are usually a non-linear function like sigmoid, tanh etc. However sometimes linear activation function works equally well. In Deep Learning context, the purpose of an activation function is to map the representation in the input space to a different space in the output. The input to each layer has an activation function , thus while learning the dictionary and the features of that layer, the updating of the features depends on the steepness (slope) of the activation function. In general it is also referred to as projection followed by selection. This introduces non-linearities that are desriable in multi-layer networks in order to detect non-linear features in the data. Typically, activation functions have a "squashing" effect.

In this thesis, The activation function used is either sigmoid or tanh for non-linear transformations .Sigmoid function maps the output between the range of [0,1] and tanh h maps the output between [-1,1]. However for few results linear tranformations worked better and were thus used.



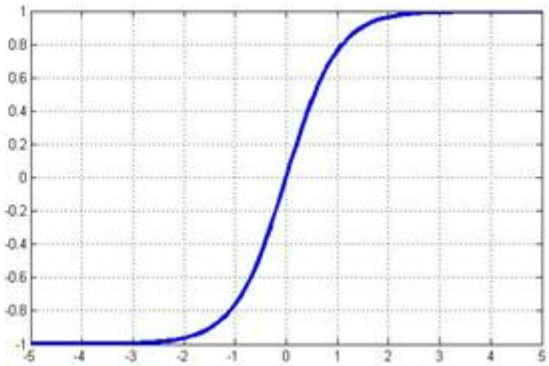


Figure 9: Sigmoid function **Tanh function**

3.4 Algorithm 1 : Deep Dictionary Learning

Equation (23) can be solved as the following Optimization Problem:

$$\min_{D_1,D_2..D_N,Z} \left\| X - D_1\Phi\left(D_2\Phi(D_3\Phi.....\Phi(D_nZ))\right)\right\|_F^2 + \lambda\left\|Z\right\|_1 \quad \ldots(24)$$

However, Equation (24) is highly non-convex and requires solving huge number of parameters. With limited amount of data, it will lead to over-fitting. To address these issues, as mentioned before, we propose a greedy approach where, we learn one layer at a time – similar to deep learning paradigm.

With the substitution, $Z_1 = \Phi\left(D_2\Phi(D_3\Phi.....\Phi(D_NZ))\right)$. Equation (24) can be written as:

$$X = D_1Z_1$$

Now this problem can be solved as single layer dictionary learning. Kindly note that the representation, Z_1, is not sparse. Hence it can be solved using alternating minimization:

$$\min_{D,Z} \left\| X - D_1Z_1\right\|_F^2 \quad \ldots(25)$$

Optimality of solving Equation (24) by alternating minimization has been proven in [56]. Therefore we follow the same approach:

$$Z_1 \leftarrow \min_{Z} \left\| X - D_1Z_1\right\|_F^2$$
$$D_1 \leftarrow \min_{D} \left\| X - D_1Z_1\right\|_F^2 \quad \ldots \text{ 25(a) \& 25(b)}$$

This is the method of optimal directions [36] and both Equation (25a) and (25b) are simple least square problems having closed form solutions.

For the second layer we substitute $Z_2 = \Phi\left(D_3\Phi(D_4\Phi.....\Phi(D_NZ))\right)$, which leads to $Z_1 = \Phi(D_2Z_2)$ or alternatively, $\Phi^{-1}(Z_1) = D_2Z_2$; this too is a single layer dictionary learning. Since the representation is dense, it can be solved using :

$$\min_{D_2,Z_2} \left\| \Phi^{-1}(Z_1) - D_2Z_2\right\|_F^2 \quad \ldots(26)$$

Equation (26) is can be solved by alternating minimization as in the case of first layer , Equation(25). Continuing in this fashion till the penultimate layer, in the final layer we have, $Z_{N-1} = \Phi(D_N Z)$ or alternatively $\Phi^{-1}(Z_{N-1}) = D_N Z$ .In the last level the sparse coding of the input to the layer can be performed, that is the coefficient, Z can be sparse. For learning sparse features, one needs to regularize by applying l_1-norm on the features. This is given by:

$$\min_{D_N, Z} \left\| \Phi^{-1}(Z_{N-1}) - D_N Z \right\|_F^2 + \lambda \left\| Z \right\|_1 \qquad \ldots(27)$$

This too is solved using alternating minimization:

$$Z \leftarrow \min_{Z} \left\| \Phi^{-1}(Z_{N-1}) - D_2 Z \right\|_F^2 + \lambda \left\| Z \right\|_1 \qquad \ldots(28(a))$$

$$D_N = \min_{D_N} \left\| \Phi^{-1}(Z_{N-1}) - D_N Z \right\|_F^2 \qquad \ldots(28(b))$$

As before, (28b) is a least square problem having a closed form solution. The solution to (28a) is although not analytic; it can be solved using the Iterative Soft Thresholding Algorithm (ISTA) [29]. The ISTA solution for (28a) is given by:

Initialize: $Z \leftarrow \min_{Z} \left\| \Phi^{-1}(Z_{N-1}) - D_2 Z \right\|_F^2$

Iterate till convergence : $B = Z + \frac{1}{\alpha} D_N^T (\Phi^{-1}(Z_{N-1}) - D_N Z)$

$$Z \leftarrow signum(B) \max(0, \left| B \right| - \frac{\lambda}{2\alpha})$$

3.4.1 Algorithm

The training algorithm [6] is as shown in Table 3.1. Note that Activation function can be linear as well as non linear.

Table 3.1

TRAINING ALGORITHM (for any activation function Φ)
Initialize D For 1st level : iterate till convergence $Z_1 \leftarrow \min_{Z} \|X - D_1 Z_1\|_F^2$ $D_1 \leftarrow \min_{D} \|X - D_1 Z_1\|_F^2$ For 2nd to penultimate level : repeat until convergence $Z_i = \min_{Z_i} \|\Phi^{-1}(Z_{i-1}) - D_2 Z_i\|_F^2$ $D_i = \min_{D_i} \|\Phi^{-1}(Z_{i-1}) - D_2 Z_i\|_F^2$ For last level : repeat until convergence $Z \leftarrow \min_{Z} \|\Phi^{-1}(Z_{N-1}) - D_2 Z\|_F^2 + \lambda \|Z\|_1$ $D_N = \min_{D_N} \|\Phi^{-1}(Z_{N-1}) - D_N Z\|_F^2$

The testing algorithm is as shown in Table 3.2. Note that the algorithms are different for linear and non-linear Activation function. It is important to note that learning multiple dictionaries cannot be collapsed into a single one even if the activation function is linear. This is because dictionary learning is bilinear. For example, if the dimensionality of the sample is m and the first dictionary is of size m x n_1 and the second one is n_1 x n_2, it is not possible to learn a single dictionary of size m x n_2 and expect the same results as a two-stage dictionary. In general, for non-linear activation functions it is not possible to collapse the multiple levels of dictionaries into a single level for testing. However, for the linear activation function, the multiple levels of dictionaries can be collapsed into a single stage by matrix multiplication of the different dictionaries and the

sparse code / features computed from the thus formed single level dictionary by standard l_1 minimization.

Table 3.2

TESTING ALGORITHM (for any linear function Φ)
Collapse D =$D_1D_2....D_N$ Compute z_{test} ,Sparse representation of test sample vector x_{test} $z_{test} \leftarrow \min_{z_{test}} \left\| x_{test} - Dz_{test} \right\|_2^2 + \lambda \left\| z_{test} \right\|_1$
TESTING ALGORITHM (for Non-linear activation function Φ)
For 1st level : Generate representation/features $z_{1,test} \leftarrow \min_{z_{1,test}} \left\| x_{test} - D_1 z_{1,test} \right\|_2^2$ For 2nd to penultimate level $z_{i,test} \leftarrow \min_{z_{i,test}} \left\| \Phi^{-1}(z_{i-1,test}) - D_i z_{i,test} \right\|_2^2$ For last level : $z_{test} \leftarrow \min_{z_{test}} \left\| \Phi^{-1}(z_{N-1,test}) - D_N z_{test} \right\|_F^2 + \lambda \left\| z_{test} \right\|_1$

3.5 Algorithm 2 : Robust Deep Dictionary Learning

We propose to add a denoising layer at the top most layer i.e layer 1 wherein the dataset is the input. The denoising layer has been added to remove impulse noise, if any, before the representations are solved for. Impulse noise is present in images as well as in Hyperspectral images in the form of Shot noise. For other noise such as Gaussian noise , the previous Algorithm works well too. However in the case of impulse noise the Root Mean Square/Euclidean(l_2) cost function does not work as well as the Absolute Deviation (l_1) cost function. The l_1 - norm is robust to measurement errors such as noise and outliers as compare to l_2 - norm . It comes in more useful when one considers the robustness of the learning model. Model robustness refers to how robustly any

learning model can handle data corruptions in prediction. Hence we try to remove the impulse noise affecting the dataset before we learn the dictionary and the coefficients. Combining l_1 and l_2 loss functions in different layers tends to give a result with fewer regression coefficients shrunk exactly to zero than in a pure L1 setting, and more shrinkage of the other coefficients.

The denoising layer is followed by the usual deep dictionary learning layers to generate features. The top most Denoising Layer can be represented as :

$$X = D_1 Z_1$$

Where $X = X_0 + N$

X_0 : Dataset

N : impulse Noise or shot noise

This can be solved using Absolute Deviation cost function as:

$$\min_{D,Z} \left\| X - D_1 Z_1 \right\|_1 \qquad \ldots(29)$$

Where $\left\| * \right\|_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} *_{i,j}$

Many methods have been proposed to solve Equation(29). Here we solve it using Split-Bregman Algorithm[54].

3.5.1 Split-Bregman Algorithm :

Assuming : $Q = X - D_1 Z_1 + B$

The problem statement gets converted to:

$$\min_{Q, D_1, Z_1} \left\| Q \right\|_1 + \mu \| Q - \left(X + D_1 Z_1 \right) - B \|_F^2$$

Split the Minimization Function:

$$P1 = \min_{Q} \left\| Q \right\|_1 + \mu \| Q - \left(X + D_1 Z_1 \right) - B \|_F^2$$

$$P2 = \min_{D_1} \| Q - \left(X + D_1 Z_1 \right) - B \|_F^2$$

$$P3 = \min_{Z_1} \| Q - \left(X + D_1 Z_1 \right) - B \|_F^2$$

Iterate till convergence : P1

P2

P3

Update Bregman variable B.

For second layer we substitute : $Z_1 = \Phi(D_2 Z_2)$ or alternatively, $\Phi^{-1}(Z_1) = D_2 Z_2$

The second layer is also a single layer dictionary learning . But this layer is solved using Euclidean cost function :

$$\min_{D_2, Z_2} \left\| \Phi^{-1}(Z_1) - D_2 Z_2 \right\|_F^2 \qquad \ldots(30)$$

Then continuing in this fashion till the penultimate layer, we impose sparsity constraint on loading coeffiecients at the last layer.

Experimental Results have shown that Algorithm 2 performs equally well as Algorithm 1 in the absence of impulse noise, but better than latter in the presence of impulse noise.

3.5.2 Algorithm

The training algorithm is as shown in Table 3.3. Note that Activation function can be linear as well as non linear.

Table 3.3

TRAINING ALGORITHM (for any activation function Φ)
Initialize D For 1st level : Assuming $: Q = X - D_1 Z_1 + B$ iterate till convergence $Q \leftarrow \min_Q \left\|Q\right\|_1 + \mu \| Q - \left(X + D_1 Z_1\right) - B \|_F^2$ $D_1 \leftarrow \min_{D_1} \| Q - \left(X + D_1 Z_1\right) - B \|_F^2$ $Z_1 \leftarrow \min_{Z_1} \| Q - \left(X + D_1 Z_1\right) - B \|_F^2$

For 2nd to penultimate level : repeat until convergence

$$Z_i = \min_{Z_i} \left\| \Phi^{-1}(Z_{i-1}) - D_2 Z_i \right\|_F^2$$

$$D_i = \min_{D_i} \left\| \Phi^{-1}(Z_{i-1}) - D_2 Z_i \right\|_F^2$$

For last level : repeat until convergence

$$Z \leftarrow \min_{Z} \left\| \Phi^{-1}(Z_{N-1}) - D_2 Z \right\|_F^2 + \lambda \left\| Z \right\|_1$$

$$D_N = \min_{D_N} \left\| \Phi^{-1}(Z_{N-1}) - D_N Z \right\|_F^2$$

The testing algorithm is as shown in Table 3.4. Note that the algorithms are different for linear and non linear Activation function.

Table 3.4

TESTING ALGORITHM (for linear activation function Φ)

For 1st level : Assuming : $R = x_{test} - D_1 z_{1,test} + b$

Generate representation/features

$$z_{1,test} \leftarrow \min_{z_{1,test}} \left\| R - (x_{test} + D_1 z_{1,test}) - b \right\|_2^2$$

$$\mathrm{R} \leftarrow \min_{R} \left\| R \right\|_1 + \mu \| R - \left(x_{test} + D_1 z_{1,test} \right) - b \|_2^2$$

Collapse D = $D_2 D_3 \ldots D_{N-1}$

Generate features for Penultimate level :

$$z_{i,test} \leftarrow \min_{z_{i,test}} \left\| z_{i-1,test} - D_i z_{i,test} \right\|_2^2$$

For last level :

$$\mathrm{z}_{test} \leftarrow \min_{z_{test}} \left\| z_{N-1,test} - D_N z_{test} \right\|_F^2 + \lambda \left\| z_{test} \right\|_1$$

TESTING ALGORITHM (for non-linear activation function Φ)

For 1st level : Assuming : $R = x_{test} - D_1 z_{1,test} + b$

Generate representation/features

$$z_{1,test} \leftarrow \min_{z_{1,test}} \left\| R - (x_{test} + D_1 z_{1,test}) - b \right\|_2^2$$

$$\mathrm{R} \leftarrow \min_{R} \left\| R \right\|_1 + \mu \| R - \left(x_{test} + D_1 z_{1,test} \right) - b \|_2^2$$

For 2nd to penultimate level

$$z_{i,test} \leftarrow \min_{z_{i,test}} \left\| \Phi^{-1}(z_{i-1,test}) - D_i z_{i,test} \right\|_2^2$$

For last level :

$$z_{test} \leftarrow \min_{z_{test}} \left\| \Phi^{-1}(z_{N-1,test}) - D_N z_{test} \right\|_F^2 + \lambda \left\| z_{test} \right\|_1$$

3.6 Going Deep in Dictionary Learning

Depth has emerged as the one of the key aspects of the learning strategies. The argument in the favour of depth is that it promotes feature re-use. We learn hierarchy of features by constructing multi level dictionaries. The pivotal idea, referred to as greedy layerwise unsupervised learning [7-10] , is to learn a hierarchy of features one level at a time, using unsupervised feature learning to learn a new transformation at each level to be composed with the previously learned transformations[7]. After greedy layerwise unsuperivised training, the resulting deep features can be used either as input to a standard machine learning predictor (such as an SVM or KNN) or as initialization for a deep supervised neural network. It was empirically observed that layerwise stacking of feature extraction often yielded better representations, e.g., in terms of classification accuracy.

Another advantage of depth is that more abstract features can be generated from less abstract features at deeper levels. The idea comes from the fact that the inputs are transformed to highly non-linear transformations at the deeper levels thus representing more variations.

The succeeding chapters containing experimental results show that sufficient depth gives better representation which is shown in terms of classification accuracy. The results show that classification performance is better when features are learnt from multi-level than single level dictionary.

The effectiveness of the proposed deep dictionary learning is evaluated on multiple benchmark databases from different areas such as images and HyperSpectral Image Classification. The results are compared with related state-of-the-art algorithms. In this work we use a linear as well as non-linear activation function depending upon whichever gives the best accuracy

The Features generated using Algorithm 1 and Algorithm 2 are sent as input to various Classifiers such as KNN , Neural Net Classifier and SVM. The Results are compared with Stats-of-the-art Machine Learning strategies such as Deep Belief Networks and Stacked Autoencoders, and also with well-known Dictionary learning algorithms such as LC-KSVD1 and LC-KSVD2.

4 | Deep Dictionary Learning on Images

We have evaluated the performance on several benchmarks datasets. They are described below :

4.1 Data Description :

4.1.1 MNIST

The MNIST dataset that consists of 28x28 images of handwritten digits ranging from 0 to 9. It is a subset of a larger set available from NIST. The dataset has 60,000 images for training and 10,000 images for testing. It should be noted that we have not performed any preprocessing on this dataset.

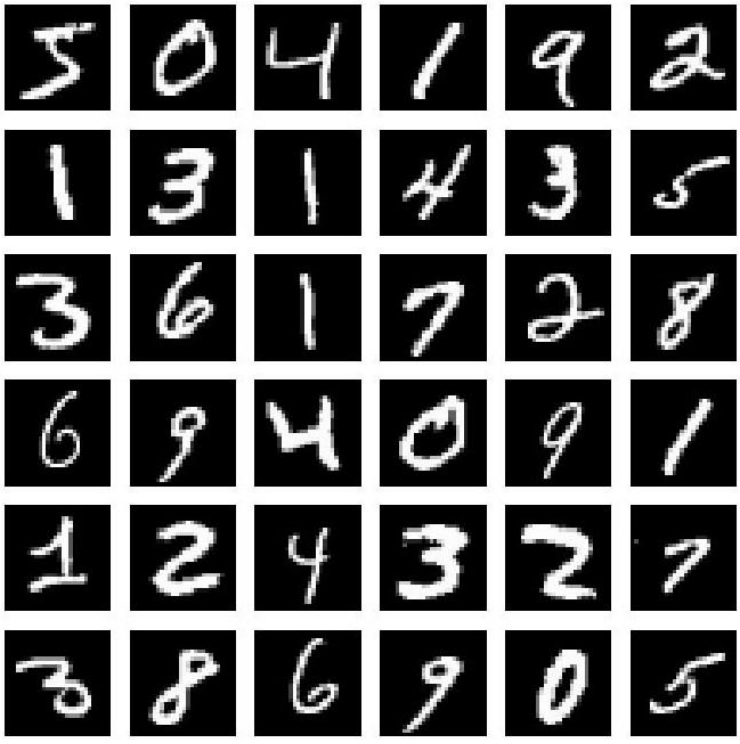


Figure 10 : MNIST dataset

Related to MNIST database, MNIST variations datasets are also used. These are more challenging databases, primarily due to fewer training samples (10,000) and larger number of test samples (50,000). The validation set of 2000 samples are not used in this work since our method does not require tuning and SAE as well as DBN are already optimized for MNIST. These were preprocessed by thresholding the pixels to 0.9.

Here is the listing of these variation databases:

1. **basic** (smaller subset of MNIST)

2. ***mnist-rot***: the digits are rotated by an angle between 0 and 2π radians. Thus the factors of variation are the rotation angle and the factors of variation already contained in MNIST, such as handwriting style.

3. ***mnist-back-rand***: a random background was inserted in the digit image. Each pixel value of the background was generated uniformly between 0 and 255.

4. ***mnist-back-image***: a patch from a black and white image was used as the background for the digit image.

5. ***mnist-rot-back-image***: the perturbations used in *mnist-rot* and *mnist-back-image* were combined

4.1.1.1 Experimental Results on MNIST and its variations:

Case 1 : Classification accuracies

Table 4.1 : Classification accuracy on MNIST and its variants for proposed methods

Dataset	DBN-3 [11]	SDAE [11]	LC KSVD1	LC KSVD2	Proposed Algorithm 1 + SVM	Proposed Algorithm 2 +KNN
MNIST 60K	98.54	98.72	93.30	93.65	98.60	98.53
MNIST Back+ Rand	93.05	89.70	87.70	87.70	92.37	92.96
MNIST Back Img	83.37	83.32	80.65	81.20	86.17	85.47
MNIST Back+ Rot	52.17	56.24	48.70	50.18	63.85	63.72
MNIST Rot digits	89.43	90.47	75.40	75.40	90.34	90.83
MNIST Basic	96.74	97.16	91.90	92.10	97.28	97.63

Case 2 : Classification accuracies in the presence of impulse noise

In this section, impulse noise was introduced in MNIST and its variations. Varying amount of pixels were made off in the entire set. This replicates the scenario where there are missing pixels in an image. The impulse noise was added randomly to individual images. Case 2 considers the case where the testing and training sets are corrupted.

It was observed that while proposed algorithms performed decently with respect to state-of-the-art methods , given that the complexity and running time is very less. Also the main aim of introducing Algorithm 2 to work with the noisy data is achieved with its performance being better than Algorithm 1.

Table 4.2 : Classification accuracy on MNIST and its variants for proposed methods with added impulse noise of 10% to training and testing sets.

Dataset	DBN	LC KSVD1	LC KSVD2	Algorithm 1 + SVM	Algorithm 2 + SVM
MNIST 60K	97.14	93.00	92.6	97.23	97.17
MNIST Back+ Rand	91.32	86.63	86.72	88.24	88.62
MNIST Back Img	79.41	79.90	78.85	80.30	80.95
MNIST Back+ Rot	50.30	47.87	46.26	55.67	57.03
MNIST Rot digits	85.95	73.92	73.47	85.74	86.13
MNIST Basic	94.72	91.8	91.32	95.25	95.47

Table 4.3 : Classification accuracy on MNIST and its variants for proposed methods with added impulse noise of 20% to both training and testing sets.

Dataset	DBN-3	LC KSVD1	LC KSVD2	Algorithm 1 + SVM	Algorithm 2 + SVM
MNIST 60K	96.48	92.52	92.73	96.79	96.87
MNIST Back+ Rand	88.20	86.00	85.77	87.29	87.50
MNIST Back Img	77.39	77.50	76.89	78.39	80.32
MNIST Back+ Rot	48.03	45.78	44.58	53.72	54.83
MNIST Rot digits	83.04	72.18	71.64	84.02	84.25
MNIST Basic	92.63	90.90	90.72	94.60	94.56

Table 4.4 : Classification accuracy on MNIST and its variants for proposed methods with added impulse noise of 30% to both training and testing sets.

Dataset	DBN	LC KSVD1	LC KSVD2	Algorithm 1 + SVM	Algorithm 2 + KNN
MNIST 60K	95.37	91.69	91.27	96.02	95.89
MNIST Back+ Rand	86.01	84.28	84.14	85.06	85.49
MNIST Back Img	75.97	76.52	76.02	76.66	78.64
MNIST Back+ Rot	43.25	43.80	43.59	50.27	51.02
MNIST Rot digits	80.48	70.96	70.51	81.54	81.75
MNIST Basic	89.64	90.04	90.28	93.42	93.28

4.1.2 USPS:

The dataset contains the numeric data obtained from the scanning of handwritten digits from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images here have been de-slanted and size normalized, resulting in 16 x 16 grayscale images of 10 classes. There are 7291 training observations and 2007 test observations. The test set is considered to be notoriously "difficult".

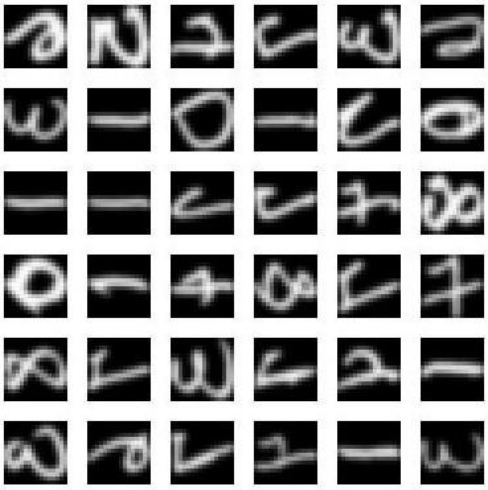


Figure 11 : USPS dataset

4.1.2.1 Experimental Results on USPS:

Case 1 : Classification accuracies

Table 4.5 : Classification accuracy on USPS for proposed methods with SVM as classifier

#	DKSVD	LCKSVD2	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
USPS	95.05	93.91	94.42	95.26	95.67	95.42

Case 2 : Classification accuracies in the presence of impulse noise

In this section, impulse noise was introduced in MNIST and its variations. Varying amount of pixels were made off in the entire set. This replicates the scenario where there are missing pixels in an image. The impulse noise was added randomly to individual images. Case 2 considers the case where the testing and training sets are corrupted. The noise was added in the range of 50-70% as with lower percentage of impulse noise there was not appreciable difference. Case 2(A) considered the case where the testing and training sets are corrupted. Case 2(B) considered that training set was noise free while the testing set was corrupted.

It was observed that while proposed algorithms performed decently with respect to state-of-the-art methods. The results for denoising case are less than LCKSVD as they were majorly designed for denoising. Also the main aim of introducing Algorithm 2 to work with the noisy data is achieved with its performance being better than Algorithm 1.

Case 2(A) : Noisy Training set & Noisy Testing set

Table 4.6 : Classification accuracy with added 50% impulse noise to training samples and testing samples for proposed methods using KNN as classifier

50% impulse noise	LCKSVD1	LCKSVD2	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
USPS	93.42	93.03	86.15	90.98	87.19	89.74

Table 4.7 : Classification accuracy with added 60% impulse noise to training samples and testing samples for proposed methods using KNN as classifier

60% impulse noise	DKSVD	LCKSVD	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
USPS	92.69	92.04	81.37	88.89	84.55	86.00

Table 4.8 : Classification accuracy with added 70% impulse noise to training samples and testing samples for proposed methods using KNN as classifier

70% impulse noise	DKSVD	LCKSVD	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
USPS	92.42	91.64	75.93	85.2	82.12	83.01

Case 2(A) : Clean Training set & Noisy Testing set

Table 4.9 : Classification accuracy with added 50% impulse noise to testing samples for proposed methods using KNN as classifier

50% impulse noise	DKSVD	LCKSVD	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
USPS	93.37	92.93	78.03	90.33	85.091	88.17

Table 4.10 : Classification accuracy with added 60% impulse noise to testing samples for proposed methods using KNN as classifier

60% impulse noise	DKSVD	LCKSVD	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
USPS	92.80	92.73	65.57	86.21	83.11	85.00

Table 4.11 : Classification accuracy with added 70% impulse noise to testing samples for proposed methods using KNN as classifier

70% impulse noise	DKSVD	LCKSVD	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
USPS	91.34	90.26	48.88	74.74	75.08	76.36

4.2 Effect of Deep Learning

This section analyzes the results of the proposed deep dictionary learning and the effect of increasing the number of layers. The features obtained via shallow dictionary learning are used for classification . The classification performance is then compared with the same number of features generated via deep dictionary learning. For MNIST shallow dictionary 50 atoms were learnt while deep dictionary learnt 300-150-50 atoms. For USPS shallow dictionary 30 atoms were learnt while deep dictionary learnt 160-60-30 atoms.

Table 4.12: Classification accuracy on USPS and MNIST and its variations for Shallow vs Deep learning for proposed methods

#	Proposed Algorithm 1		Proposed Algorithm 2	
	Shallow [50]	**Deep [300-150-50]**	**Shallow [50]**	**Deep [50]**
MNIST 60K	93.75	98.67	93.75	98.53
MNIST Back+ Rand	87.19	92.38	87.19	92.96
MNIST Back Img	78.86	86.17	78.86	85.47
MNIST Back+ Rot	54.40	63.85	54.40	63.72
MNIST Rot digits	84.19	90.344	84.19	90.83
MNIST Basic	95.02	97.28	95.02	97.63
	Shallow [30]	**Deep [160-60-30]**	**Shallow [30]**	**Deep [150-70-30]**
USPS	92.64	95.017	92.64	95.017

5 | Deep Dictionary Learning on HyperSpectral Images

Hyperspectral data classification is a hot topic in remote sensing community. In recent years, significant effort has been focused on this issue. However, most of the methods extract the features of original data either in a shallow manner or uses Convolutional Neural Net (CNN), Deep Belief Network(DBN) and Stacked Autoencoder (SAE) for deep learning. SAE was introduced for this problem in [55] by Chen et al; the same researchers applied DBN to the problem to publish another paper [56]. There are a two papers that use CNN for the said problem [57,58]; the main problem of using CNN for this problem is that it is data hungry and yields poor results when the training data is limited. In this chapter, we introduce our deep dictionary learning approach into hyperspectral image classification. The proposed approaches are used on the raw hyperspectral datasets as well as their pre-processed versions. Experimental results with hyperspectral data indicate that the proposed algorithms using raw data performed more or less at par with the State-of-the-art methods using preprocessed data especially for limited amount of training data(practical scenario). The complexity and time taken is also way less. The proposed algorithm learns representation of the well-known hyperspectral scenes in an unsupervised manner. Few experiments were also conducted in the presence of shot noise and it was observed that while the classification accuracies were little short of the State-of-the-art methods, Algorithm 2 outperforms Algorithm 1. In addition, this reveals that deep learning system has huge potential for hyperspectral data classification.

5.1 HyperSpectral Data

Hyperspectral data, also known as hyperspectral datacube, is a 3 dimensional data, produced by instruments called imaging spectrometers. This technology combines the power of digital imaging and spectroscopy. For each pixel in an image, a hyperspectral camera acquires the light intensity (radiance) for a large number (typically a few tens to several hundred) of contiguous spectral bands. Every pixel in the image thus contains a continuous spectrum (in radiance or reflectance) and can be used to characterize the objects in the scene with great precision and detail.

Hyperspectral imaging divides the spectrum into numerous bands. This technique of dividing images into bands can be extended beyond the visible providing much more detailed information about a scene than a normal color camera which captures a scene in three bands (red, green, and blue), In hyperspectral imaging, the recorded spectra have fine wavelength resolution and cover a wide range of wavelengths. Hence, hyperspectral imaging leads to a vastly improved ability to classify the objects in the scene based on their spectral properties.

Hyperspectral sensors look at objects using a vast portion of the electromagnetic spectrum. Certain objects leave unique 'fingerprints' in the electromagnetic spectrum. Known as spectral signatures, these 'fingerprints' enable identification of the materials that make up a scanned object.

Figuratively speaking, hyperspectral sensors collect information as a set of 'images'. Each image represents a narrow wavelength range of the electromagnetic spectrum, also known as a spectral band. These 'images' are combined to form a three-dimensional (x,y,λ) hyperspectral data cube for processing and analysis, where x and y represent two spatial dimensions of the scene, and λ represents the spectral dimension (comprising a range of wavelengths).

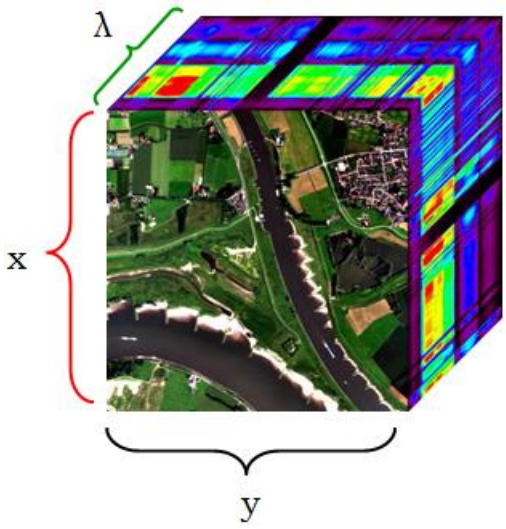


Figure 12: Hyperspectral Datacube

Hyperspectral cubes are generated from airborne sensors like the NASA's Airborne Visible/Infrared Imaging Spectrometer(AVIRIS), or from satellites like NASA's EO-1 with its hyperspectral instrument Hyperion.

Recent advances in sensor design and processing speed has cleared the path for a wide range of applications employing hyperspectral imaging, ranging from satellite based/airborne remote sensing and military target detection to industrial quality control and lab applications in medicine and biophysics. Due to the rich information content in hyperspectral images, they are uniquely well suited for automated image processing, whether it is for online industrial monitoring or for remote sensing.

5.2 Dataset Description

5.2.1 Indian Pines

The dataset consists of a scene of 145X145 pixels and 224 spectral reflectance bands in the wavelength range 0.4–2.5 micro meters. It was gathered by AVIRIS sensor over the Indian Pines test site in North-western Indiana .The Indian Pines scene contains two-thirds agriculture, and one-third forest or other natural perennial vegetation.. Since the scene is taken in June some of

the crops present, corn, soybeans, are in early stages of growth with less than 5% coverage. The ground truth available is designated into sixteen classes and is not all mutually exclusive. Class zero represents the background and accounts for almost 50% of the data. Rest 50% pixels are composed of remaining 16 classes. The number of bands were reduced to 200 by removing bands number: [104-108], [150-163], 220 covering the region of water absorption. Hence the final dataset used is of 145X145 pixels with 200 spectral bands[59]. The dataset and detailed description is given in [59].

Table 5.1 provides with the groundtruth classes along with the sample number of each class.

Table 5.1

#	**Class**	**Samples**
1	Alfalfa	46
2	Corn-notill	1428
3	Corn-mintill	830
4	Corn	237
5	Grass- Pasture	483
6	Grass- trees	730
7	Grass- Pasture - mowed	28
8	Hay- windrowed	478
9	Oats	20
10	Soyabean-notill	972
11	Soyabean-mintill	2455
12	Soyabean-clean	593
13	Wheat	205
14	Woods	1265
15	Buildings-Grass-Trees-Drives	386
16	Stone-Steel-Towers	93

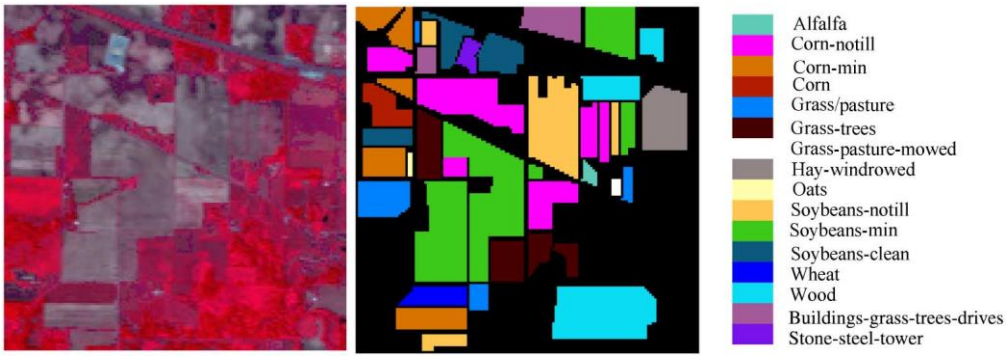


Figure 13 : a) False color image of Indian Pines (band 50,27,17)
b) Ground truth representing 16 crop cases

5.2.2 Pavia University

This scene was acquired by the Reflective Optics System Imaging Spectrometer (ROSIS) during a flight campaign over Pavia, nothern Italy. Pavia University is 610*610 pixels, but some of the samples in image contain no information and have to be discarded before the analysis. Thus the final scene is of 340*610 pixels. 115 bands were collected over 0.43 - 0.86 μm range of the electromagnetic spectrum. In the experiment, some bands were removed due to noise; the remaining 103 bands were used for the classification The spatial resolution is 1.3 m. Image groundtruth consists of 9 classes. Class zero represents background and accounts for almost 50% of the data. Rest 50% pixels are composed of remaining 9 classes. The dataset and detailed description is given in [59].

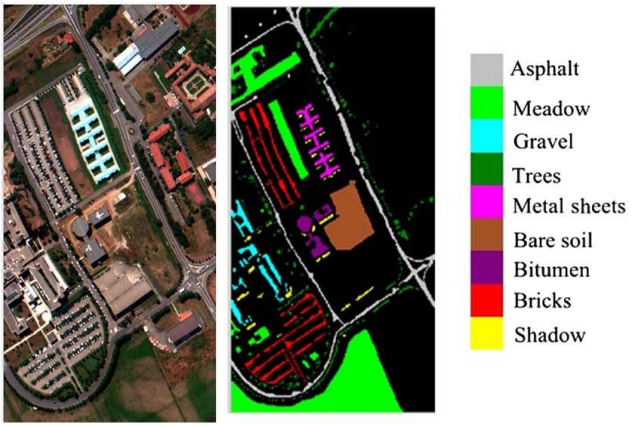


Figure 14: a) Pavia, Italy. False-color composite (Band 10, 27, 46)
b) Groundtruth representing 9 land-cover classes.

Table 5.2 provides with the groundtruth classes along with the sample number of each class.

Table 5.2

#	Class	Samples
1	Asphalt	6631
2	Meadows	18649
3	Gravel	2099
4	Trees	3064
5	Painted metal sheets	1345
6	Bare soil	5029
7	Bitumen	1330
8	Self-Blocking bricks	3682
9	Shadows	947

5.3 Input Features

A) Spectral Features

Input X is, generally, the raw spectral data collected as a one dimensional (1-D) vector for each pixel. That is we take the responses of all the spectral channels into the input space.

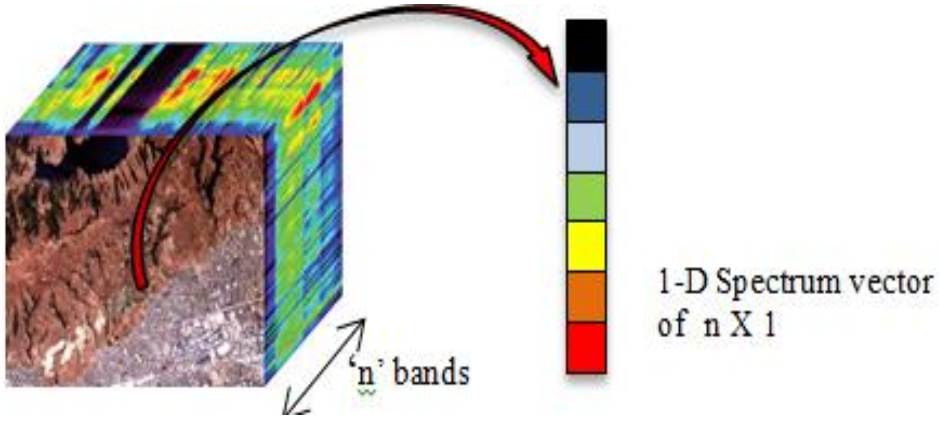


Figure 15: Spectral vector of Hyperspectral Features

Then, proposed DDL algorithms are applied to learn the representative and robust features from the inputs via several layers of linear/nonlinear feature transformation. Finally, the learned features are input to a classifier to produce the class labels in the absence of any noise for Case 1 or in the presence of varying amount of shot noise for Case 2.

B) Spectral –Spatial Features

For this we integrate the spectral and spatial features together to construct a spectral–spatial-based classification framework. Pure spectral features and spatial features both provide a discriminating power for the pixel-wise classification. The spectral feature of a pixel contains important information for discriminating different kinds of ground categories.

The Spatial vector with the spatial information of neighbourhood of a pixel , decreases the intra-class variance.

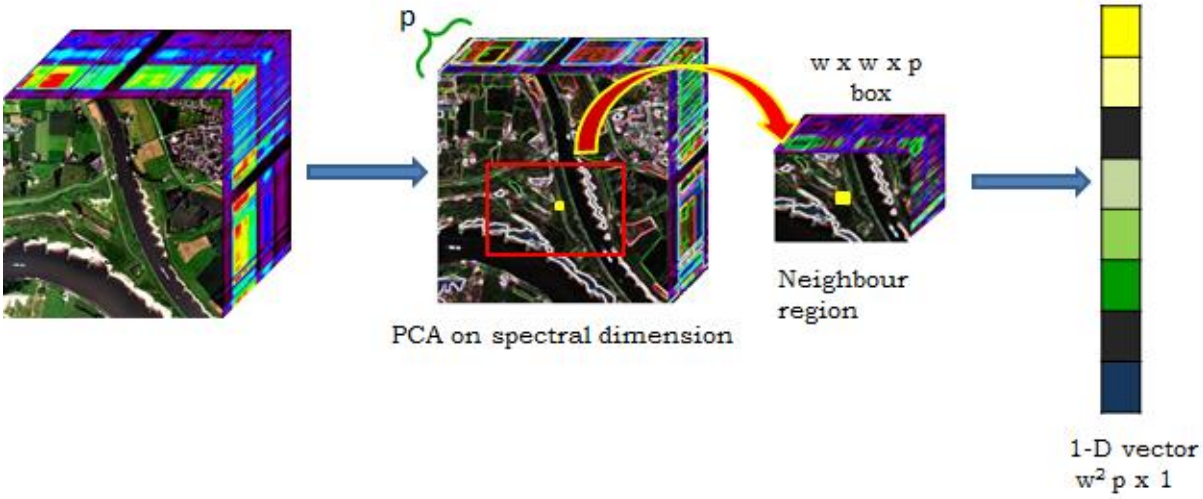


Figure 16: Spatial Vector of Hyperspectral Features

Forming a hybrid set of spectral–spatial features can present more reliable classification. The integration of multiple features is addressed by using a vector stacking (VS) approach. That is to say, for each pixel, spatial vector is added to the end of the spectral vector.

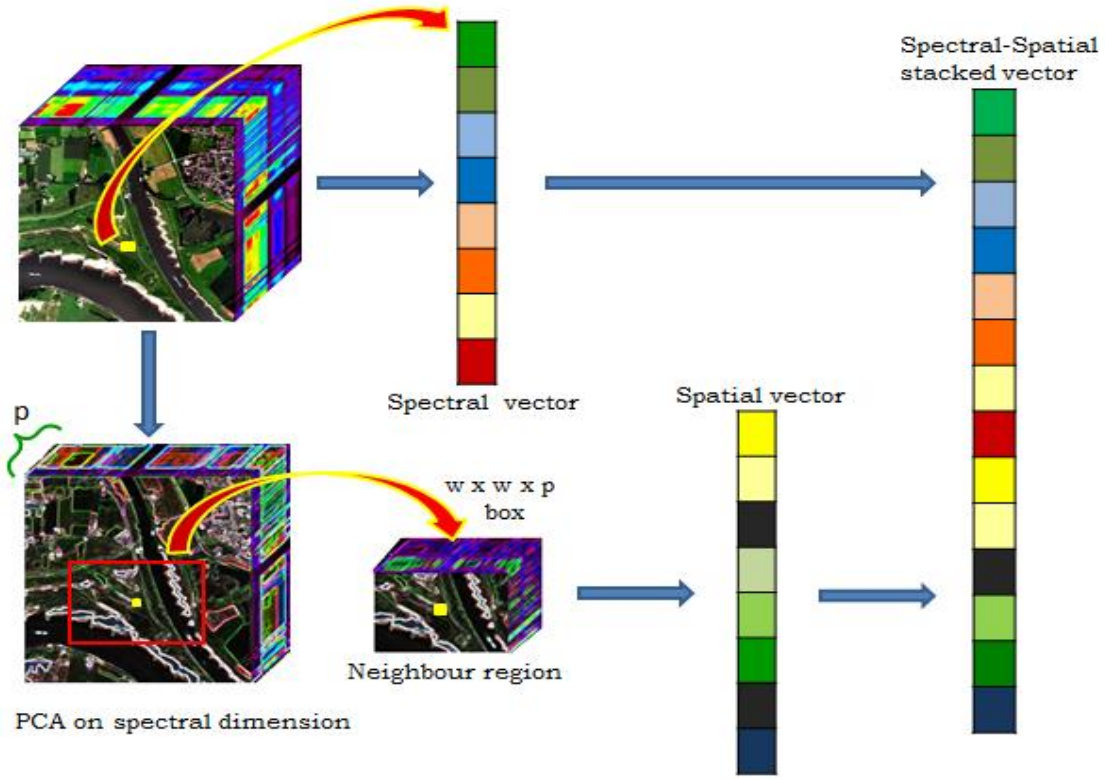


Figure 17 : Spectral-Spatial Vector of Hyperspectral Features

Then, proposed DDL algorithms are applied to learn the representative and robust features from the inputs via several layers of linear/nonlinear feature transformation. Finally, the learned features are input to a classifier to produce the class labels in the absence of any noise for Case 1 or in the presence of varying amount of shot noise for Case 2.

5.4 Experimental Results

Prior studies on deep learning based classification assumed an overtly optimistic scenario [55,56] – they assumed 50% (30% training + 20% validation) labelled data is available; and only 50% need to be predicted. In

this work we consider a more realistic scenario; we assume only 10% or 20% of the each of the labelled class available. No validation set was used.
The results were compared with Stacked Autoencoder (SAE) and Deep Belief Network(DBN). The implementation for the SAE has been obtained from [56] for Case 1 with the [training: validation: testing] set being [1:1:8] , for rest of the cases i.e Case 1(b) and Case 2 with varying ratios of testing and training set and for DBN ,the results were generated from the rasmusbergpalm/DeepLearnToolbox [60]. Overall Accuracy (OA) is used to indicate the classification accuracy.

In this section, we evaluate our proposed DDL-based framework .The feature selection architecture used for input to proposed algorithms are the spectral features.

Case 1 : Classification accuracies in the absence of noise

Case 1(a) : The accuracies for DBN and SAE were calculated from [60] as the code released for SAE[55] does not include the case for validation set being zero for [training: testing] cases being [1:9] respectively. The case for validation and training being 5% each could not be considered as well, because the set ratio were to be in integers.

Table 5.3 : Classification accuracy with 10% training samples and 90% testing samples for proposed methods using Neural Net as classifier

10% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	**79.48**	**83.00**	**81.76**	**82.07**
Pavia University	**90.37**	**92.16**	**93.52**	**93.15**

Case 1(b) : For the below result, the accuracies for SAE were generated from the code released online by authors of paper [55].

Table 5.4 :Classification accuracy with 20% training samples and 80% testing samples for proposed methods using Neural Net as classifier

20% Training samples	DBN	SAE [55]	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	**80.32**	**85.86**	**87.19**	**86.82**
Pavia University	**94.94**	**96.32**	**96.30**	**97.68**

Case 2 : Classification accuracies in the presence of shot noise

For this section, Shot noise was introduced in the Hyperspectral images. Varying amount of pixels were made off throughout their respective wavelengths. This replicates the scenario where one or more sensors are not working while capturing the image, thus having no information throughout their respective wavelength bands. 10%,20% and 30% of shot noise was introduced for below explained two cases. Case 2(A) considered the case where the testing and training sets are corrupted. Case 2(B) considered that training set was noise free while the testing set was corrupted.

It was observed that while proposed algorithms performed decently with respect to state-of-the-art methods , the main aim of introducing Algorithm 2 to work with the noisy data is achieved with its performance being better than Algorithm 1.

Case 2(A) : Noisy Training set & Noisy Testing set

i) 10% shot noise

Table 5.5 :Classification accuracy with added noise to 10% training samples and 90% testing samples for proposed methods using Neural Net as classifier

10% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	74.83	77.22	78.62	78.93
Pavia University	85.89	75.36	85.84	86.00

Table 5.6 : Classification accuracy with added noise to 20% training samples and 80% testing samples for proposed methods using Neural Net as classifier

20% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	79.64	82.24	81.45	82.69
Pavia University	86.06	83.72	88.21	88.67

ii) 20% shot noise

Table 5.7 : Classification accuracy with added noise to 10% training samples and 90% testing samples for proposed methods using Neural Net as classifier

10% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	72.81	73.57	74.70	76.28
Pavia University	80.62	74.54	81.09	82.73

Table 5.8 : Classification accuracy with added noise to 20% training samples and 80% testing samples for proposed methods using Neural Net as classifier

20% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	75.96	78.41	77.63	79.01
Pavia University	80.502	81.33	83.50	84.65

iii) 30% shot noise

Table 5.9 : Classification accuracy with added noise to 10% training samples and 90% testing samples for proposed methods using Neural Net as classifier

10% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	65.42	60.82	71.48	73.09
Pavia University	74.71	72.93	75.35	76.60

Table 5.10 : Classification accuracy with added noise to 20% training samples and 80% testing samples for proposed methods using Neural Net as classifier

20% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	71.08	74.66	75.82	77.35
Pavia University	75.38	77.31	78.25	79.16

Case 2(B) : Clean Training set & Noisy Testing set

i) 10% shot noise

Table 5.11 : Classification accuracy with 10% training samples and added noise to 90% testing samples for proposed methods using Neural Net as classifier

10% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	78.00	79.35	79.78	78.84
Pavia University	80.91	75.66	81.91	81.08

Table 5.12 : Classification accuracy with 20% training samples and added noise to 80% testing samples for proposed methods using KNN as classifier

20% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	76.95	83.92	83.43	84.01
Pavia University	83.30	82.96	85.37	86.93

ii) 20% shot noise

Table 5.13 : Classification accuracy with 10% training samples and added noise to 90% testing samples for proposed methods using KNN as classifier

10% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	73.26	75.27	78.32	78.57
Pavia University	72.64	70.69	73.07	74.10

Table 5.14 : Classification accuracy with 20% training samples and added noise to 80% testing samples for proposed methods using KNN as classifier

20% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	73.33	80.00	80.19	82.26
Pavia University	74.45	77.15	75.26	76.88

iii) 30% shot noise

Table 5.15 : Classification accuracy with 10% training samples and added noise to 90% testing samples for proposed methods using KNN as classifier

10% Training samples	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	62.19	66.38	74.74	76.84
Pavia University	65.16	63.67	64.17	70.16

30% shot noise ....(cont.)

Table 5.16 : Classification accuracy with 20% training samples and added noise to 80% testing samples for proposed methods using KNN as classifier

20% Training samples + 30% shot noise	DBN	SAE	Proposed Algorithm 1	Proposed Algorithm 2
Indian Pines	67.61	75.48	81.45	82.01
Pavia University	68.72	73.34	69.98	72.29

5.5 Effect of Deep Learning

This section analyzes the results of the proposed deep dictionary learning and the effect of increasing the number of layers. The features obtained via shallow dictionary learning are used for classification . The classification performance is then compared with the same number of features generated via deep dictionary learning.

Table 5.17: Classification accuracy on Indian Pines for Shallow vs Deep learning for proposed methods

Indian Pines	10% training data Shallow (41)	10% training data Deep (163-82-41)	20% training data Shallow (41)	20% training data Deep(163-82-41)
Proposed Algorithm1	71.76	81.75	81.67	87.19
Proposed Algorithm 2	71.76	82.07	81.67	86.82

Table 5.18: Classification accuracy on Pavia University for Shallow vs Deep learning for proposed methods

Pavia University	10% training data Shallow (20)	10% training data Deep (210-50-20)	20% training data Shallow (35)	20% training data Deep (300-150-35)
Proposed Algorithm1	86.88	93.52	92.52	96.30
Proposed Algorithm 2	86.88	93.15	92.52	97.68

Graphs for Classification Accuracies for Case 2(A)

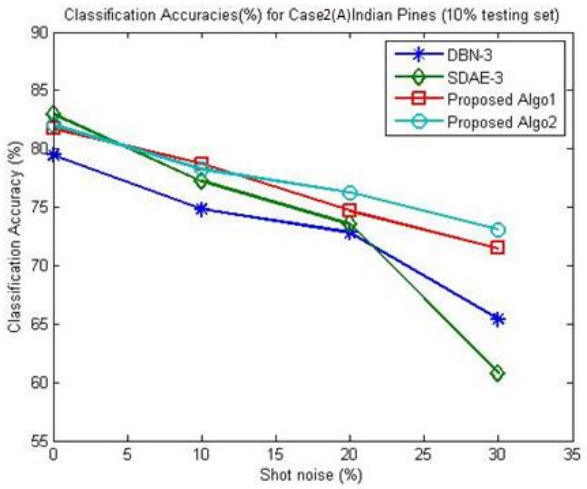


Figure 18: Classification accuracy for Indian Pines (10% Test set)

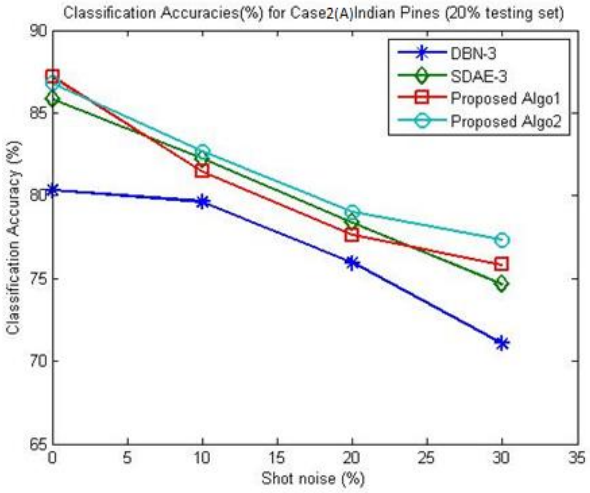


Figure 19: Classification Accuracy for Indian Pines (20% Test set)

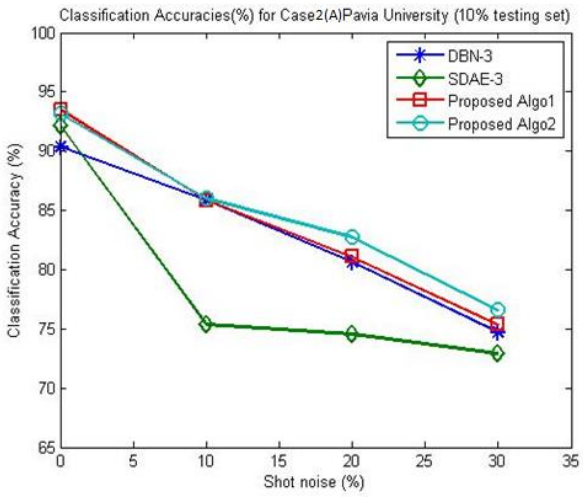


Figure 20: Classification accuracy for Pavia U (10% Test set)

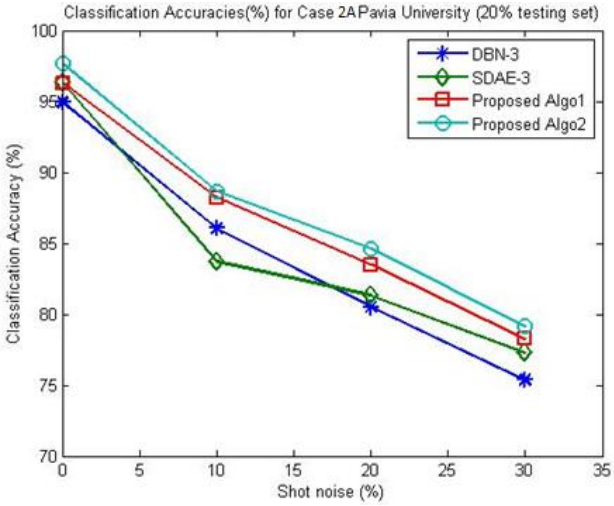


Figure 21 : Classification Accuracy for Pavia U (20% Test set)

Graphs for Classification Accuracies for Case 2(B)

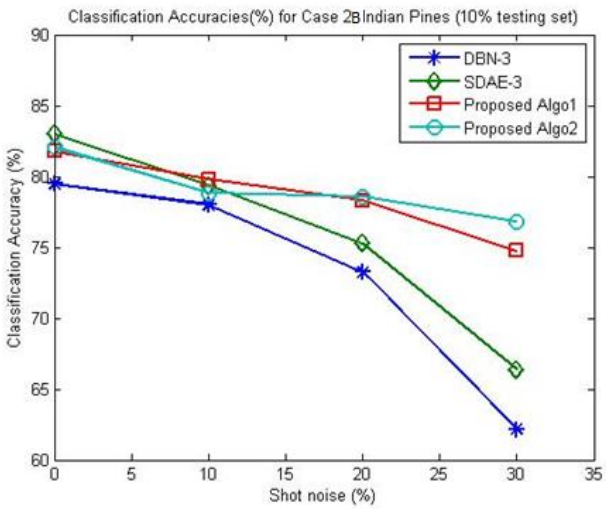


Figure 22 : Classification Acuracy for Indian Pines (10% Test Set)

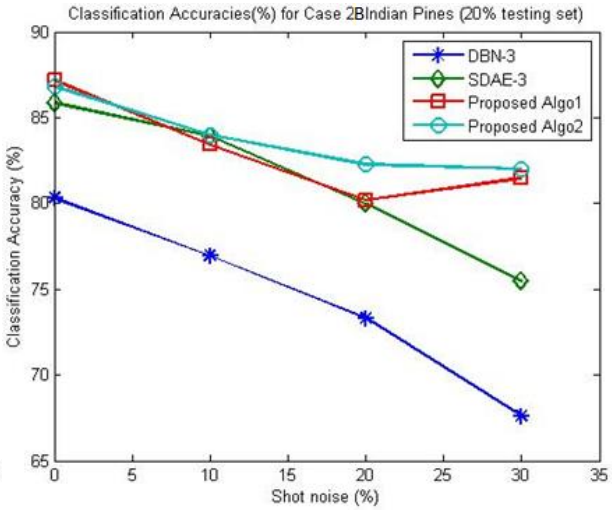


Figure 23: Classification Acuracy for Indian Pines (20% Test set)

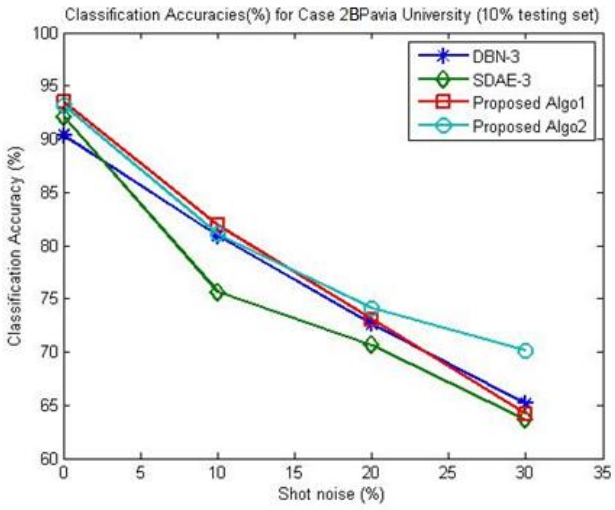


Figure 24: Classification Accuracy for Pavia U (10% Test set)

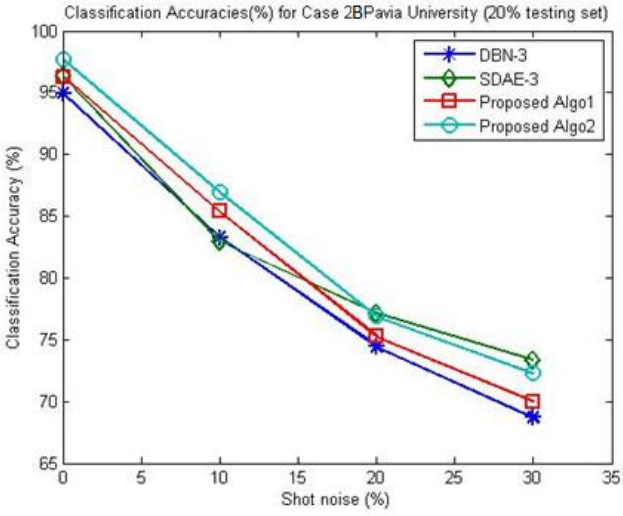


Figure 25 : Classification Accuracy for Pavia U (20% Test set)

Bibliography

1. K. Rao and P. Yip. Discrete Cosine Transform: Algorithms, Advantages and Applications. Academic Press, 1990.
2. S. Mallat. A Wavelet Tour of Signal Processing: The Sparse Way. Academic Press, third edition, 2008.
3. I. Daubechies. Ten Lectures on Wavelets. SIAM, 1992.
4. G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks", Science, Vol. 313 (5786), pp. 504-507, 2006.
5. H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition". Biologic Cybernetics, Vol. 59 (4–5), pp. 291–294, 1989.
6. S. Tariyal, A. Majumdar, R. Singh, M. Vatsa, “Greedy Deep Dictionary Learning”, arXiv:1602.00203.
7. Y. Bengio, A. Courville and P. Vincent, "Representation Learning: A Review and New Perspectives," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798-1828, Aug. 2013.
8. Y. Bengio, P. Lamblin, P. Popovici and H. Larochelle, “Greedy Layer-Wise Training of Deep Networks”, Advances in NeuralInformation Processing Systems, 2007.
9. G. E. Hinton, S. Osindero and Y. W. Teh, “A fast learning algorithm for deep belief nets”, Neural Computation, Vol. 18, pp. 1527-1554,2006.
10. Y. Bengio, “Learning deep architectures for AI”, Foundations and Trends in Machine Learning, Vol. 1(2), pp. 1-127, 2009.
11. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P. A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a

deep network with a local denoising criterion", Journal of Machine Learning Research, Vol. 11, pp. 3371-3408, 2010.

12. M. Elad and M. Aharon, "Image Denoising Via Learned Dictionaries and Sparse representation," 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006, pp. 895-900.

13. G. Peyre, "Texture Synthesis with Grouplets," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 4, pp. 733-746, April 2010.

14. M. Zibulevsky and B. A. Pearlmutter, "Blind Source Separation by Sparse Decomposition in a Signal Dictionary," in Neural Computation, vol. 13, no. 4, pp. 863-882, April 1 2001.

15. J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In Proceedings of the International Conference on Machine Learning (ICML), 2009a.

16. K. Engan, S. Aase, and J. Hakon-Husoy, "Method of optimal directions for frame design," IEEE International Conference on Acoustics, Speech, and Signal Processing, 1999.

17. E. J. Candes and D. L. Donoho, "Recovering edges in ill-posed inverse problems: Optimality of curvelet frames," Ann. Statist., vol. 30, no. 3, pp. 784–842, Jun. 2002.

18. E. J. Candès and D. L. Donoho, "New tight frames of curvelets and the problem of approximating piecewise C images with piecewise C^2 edges," Commun. Pure Appl. Math., vol. 57, pp. 219–266, Feb. 2004.

19. D. L. Donoho, "Wedgelets: Nearly minimax estimation of edges," Ann. Statist., vol. 27, no. 3, pp. 859–897, Jun. 1998.

20. Moritz Hardt, "Understanding Alternating Minimization for Matrix Completion" , arXiv:1312.0925 [cs.LG]

21. R. Rubinstein, A. M. Bruckstein and M. Elad, "Dictionaries for Sparse Representation Modeling", Proceedings of the IEEE, Vol. 98(6), pp. 1045-1057, 2010.

22. M. Aharon, M. Elad and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation", IEEE Transactions on Signal Processing, Vol. 54 (11), pp. 43114322, 2006.

23. J. A. Tropp and A. C. Gilbert, "Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit," in IEEE Transactions on Information Theory, vol. 53, no. 12, pp. 4655-4666, Dec. 2007.

24. B. K. Natarajan, "Sparse approximate solutions to linear systems", SIAM Journal on computing, Vol. 24, pp. 227-234, 1995.

25. D. Needell, J.A. Tropp, "COSAMP: Iterative signal recovery from incomplete and inaccurate samples" . Appl. Comput. Harmon. Anal., 26 (3) (2009), pp. 301–321

26. D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined linear equations by stagewise Orthogonal Matching Pursuit (StOMP)," IEEE Transactions on Information Theory, Volume 58 Issue 2, February 2012, pp 1094-1121.

27. R. Tibshirani , "Regression Shrinkage and selection via LASSO" , Journal of royal statistical society, Series B (methodological), Volume 58 , Issue 1,1996 ,pp 267-288.

28. Beck, A., Teboulle, M., " A fast iterative shrinkage-thresholding algorithm for linear inverse problems". SIAM Journal on Imaging Sciences 2(1), 183–202 (2009)

29. I. Daubechies, M. Defriese, and C. De Mol. "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint" . Commun. Pure Appl. Math, LVII:1413–1457, 2004.

30. E. Cande`s, M. Wakin, and S. Boyd, "Enhancing sparsity by reweighted '1 minimization," J. Fourier Anal. Appl., vol. 14, no. 5–6, pp. 877–905, 2008.

31. S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," IEEE Trans. Signal Process., vol. 41, no. 12, pp. 3397–3415, 1993

32. I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: A re-weighted norm minimization algorithm," IEEE Trans. Signal Process., vol. 45, pp. 600–616, 1997

33. Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in Conf. Rec. 27th Asilomar Conf. Signals, Syst. Comput., 1993, vol. 1

34. S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," SIAM Rev., vol. 43, no. 1, pp. 129–159, 2001

35. D. L. Donoho and I. M. Johnstone, "Ideal denoising in an orthonormal basis chosen from a library of bases," Comp. Rend. Acad. Sci., ser. A, vol. 319, pp. 1317–1322, 1994.

36. R. Coifman and D. L. Donoho, "Translation invariant denoising," in Wavelets and Statistics. New York: Springer-Verlag, 1995, vol. 103, Lecture Notes in Statistics, pp. 120–150.

37. J. L. Starck, M. Elad, and D. L. Donoho, "Image decomposition: Separation of texture from piece-wise smooth content," in SPIE Conf. Signal Image Process.: Wavelet Applicat. Signal Image Process. X, SPIE 48th Annu. Meeting, San Diego, CA, Aug. 3–8, 2003.

38. J. L. Starck, E. J. Candes, and D. L. Donoho, "The curvelet transform for image denoising," IEEE Trans. Image Process., vol. 11, pp. 670–684, 2002.

39. M. Elad, J. L. Starck, P. Querre, and D. L. Donoho, “Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA),” J. Appl. Comput. Harmon. Anal., submitted for publication

40. J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. “Robust face recognition via sparse representation”. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 31, 2, pp. 210–227, 2009.

41. Q. Zhang and B. Li, "Discriminative K-SVD for dictionary learninin face recognition". IEEE Conference of Computer Vision anPattern Recognition, 2010.

42. Z. Jiang, Z. Lin and L. S. Davis, "Learning A Discriminative Dictionary for Sparse Coding via Label Consistent K-SVD", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 35, pp. 2651-2664, 2013.

43. H. Larochelle and Y. Bengio, “Classification using Discriminative Restricted Boltzmann Machines”, International Conference on Machine Learning, 2008.

44. Z. Cui, S. S. Ge, Z. Cao, J. Yang and H. Ren, “Analysis of Different Sparsity Methods in Constrained RBM for Sparse Representation in Cognitive Robotic Perception”, Journal of Intelligent Robot and Systems, pp. 1-12, 2015.

45. H. Luo, R. Shen and C. Niu, “Sparse Group Restricted Boltzmann Machines”, arXiv:1008.4988v1 .

46. R. Salakhutdinov and G. Hinton, “Deep Boltzmann Machines”, International Conference on Artificial Intelligence and Statistics, 2009.

47. L.I.Smith. (2002, February 26). A tutorial on principal components analysis. Available: http://csnet.otago.ac.nz/cosc453/student_tutorials/principal_compone n ts.pdf

48. Ricardo Gutierrez-Osuna, datajobs.com/data-science-repo/LDA-Primer-[Gutierrez-Osuna].pdf

49. W. Cheung and G. Hamarneh, "n-SIFT: n -Dimensional Scale Invariant Feature Transform," in IEEE Transactions on Image Processing, vol. 18, no. 9, pp. 2012-2021, Sept. 2009.

50. D. Huang, C. Shan, M. Ardabilian, Y. Wang and L. Chen, "Local Binary Patterns and Its Application to Facial Image Analysis: A Survey," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 6, pp. 765-781, Nov. 2011

51. P. Jain, P. Netrapalli and S. Sanghavi, "Low-rank Matrix Completion using Alternating Minimization", Symposium on Theory Of Computing, 2013

52. C. Hillar and F. T. Sommer, "When can dictionary learning uniquely recover sparse data from subsamples?", arXiv:1106.3616v3.

53. D. A. Spielman, H. Wang and J. Wright, "Exact Recovery of Sparsely-Used Dictionaries", International Conference On Learning Theory, 2012

54. Tom Goldstein , Stanley Osher, The Split Bregman Method for L1-Regularized Problems, SIAM Journal on Imaging Sciences, v.2 n.2, p.323-343, April 2009

55. Y. Chen, Z. Lin, X. Zhao, G. Wang and Y. Gu, "Deep Learning-Based Classification of Hyperspectral Data", in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 7, no. 6, pp. 2094-2107, June 2014.

56. Y. Chen, X. Zhao and X. Jia, "Spectral–Spatial Classification of Hyperspectral Data Based on Deep Belief Network," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 8, no. 6, pp. 2381-2392, June 2015.

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

58. W. Hu, Y. Huang, L. Wei, F. Zhang and L. Hengchao, "Deep Convolutional Neural Networks for Hyperspectral Image Classification", Hindawi Journal of Sensors, Vol. 2015.

59. http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes

60. https://github.com/rasmusbergpalm/DeepLearnToolbox

61. A. Makhzani and B. Frey, "k-Sparse Autoencoders", arXiv:1312.5663, 2013.

62. K. Cho, "Simple sparsification improves sparse denoising autoencoders in denoising highly noisy images", International Conference on Machine Learning, 2013