Static Hand Gesture Recognition using Stacked Denoising Sparse Autoencoders

Varun Kumar, G.C. Nandi, Rahul Kala
Robotics And Artificial Intelligence Laboratory
Indian Institute of Information Technology, Allahabad
India
Email : varun.k.iiit@gmail.com

Citation: V. Kumar, G. C. Nandi and R. Kala, "Static hand gesture recognition using stacked Denoising Sparse Autoencoders," 2014 Seventh International Conference on Contemporary Computing (IC3), Noida, 2014, pp. 99-104. **Full Text Available At:** https://ieeexplore.ieee.org/document/6897155

***Abstract*—With the advent of personal computers, humans have always wanted to communicate with them in either their natural language or by using gestures. This gave birth to the field of Human Computer Interaction and its subfield Automatic Sign Language Recognition. This paper proposes the method of automatic feature extraction of the images of hand. These extracted features are then used to train the Softmax classifier to classify them into 20 classes. Five stacked Denoising Sparse Autoencoders (DSAE) trained in unsupervised fashion are used to extract features from image. The proposed architecture is trained and tested on a standard dataset [1] which was extended by adding random jitters such as rotation and Gaussian noise. The performance of the proposed architecture is 83% which is better than shallow Neural network trained on manual hand-engineered features called Principal components which is used as a benchmark.**

Keywords : Static hand gesture recognition, Deep learning, Autoencoders.

I. Introduction

Computers are becoming more and more ubiquitous day by day. And to control them just by using gestures make them more acceptable to a wide range people especially illiterate people. A gesture can be defined as physical movement of hands, arms, face and body with the intent to convey information or meaning [2]. Gestures recognition has various applications such as desktop interaction [3], virtual reality [4], teaching robots using imitation learning [5]. With the availability of RGB-D cameras (i.e cameras that provide depth information along with RGB color information) of good performance-to-cost ratio such as Kinect, there has been a lot of research in the field of gesture recognition. There are basically 2 technologies that enable the hand recognition, namely contact based and vision based devices. Contact based devices such as haptic gloves are based on physical contact with the user and therefore are intrusive in nature so they are not so famous. Vision based technologies are more famous and challenging. They involve the use of cameras (both monocular and stereo based. Vision based technologies are more challenging because they depend on the images acquired from the camera which in low illumination condition fail.

A complete hand gesture based human computer interaction system using vision involves 3 stages. First is detecting the hand, second is tracking the hand and the third is recognition of gesture performed by it. This paper deals with the recognition part of the system.

In general there are two broad categories under which we can categorize the vision based hand gesture representation. First is the 3D model based methods and second is appearance based methods [7]. 3D model based methods are computationally expensive, hence are not used in real time applications [8]. Therefore in this paper 2D appearance based model was used for classification.

Several methods for the purpose of hand gesture detection and recognition have been proposed. Triesh and Malsburg in their paper [9] uses elastic graph matching for the purpose of recognition of hand.

Another feature incorporated to recognize the hand gesture is the shape of the hand. The contour gives the shape of the hand. Benefit of using shape based model is that it is independent of the illumination and also skin color. Contour is extracted using edge detector therefore if the background is complex, this may result in a lot of irrelevant edges and contours. Therefore complex pre-processing is needed to reduce the effect of these extra edges. [11] deals with using Fourier descriptors to recognize the gestures. [12] uses Hu-moments as features. As the test sample is compared with all data samples, K-nearest neighbor would be computationally expensive if the size of data is very large. Another contour based method is to count the number of finger tips. Each finger tip shows as a peak in the contour. Then counting the number of peaks gives the number of finger tips. This method is very simple but it does not scale to complex gestures. Another method of classifying simple gestures is to calculate the convexity defects [13] in the hand image. The one added to the number of convexity defects gives the number of fingertips. Contour based methods are simple but not very famous because of the issue of self occlusion. There are various complex hand gestures that involves self occlusion. Because of it, the geometry of the contour changes hence the classification fails.

Another work [14] employs Support Vector Machine (SVM) for the purpose of hand gesture classification.

This paper proposes an architecture to automatically extract the features from the raw image data. This architecture is composed of Denoising Sparse Autoencoders stacked on top of each other and trained on the unlabelled data of hand images.

Then use these features to train multinomial logistic (Softmax) classifier to classify static hand gestures.

A standard dataset [1] consisting of images of static hand gestures was used. It was extended by adding random jitters and then it was used for training and testing.

II. Methodology

For the purpose of hand gesture recognition, stacked Denoising Sparse autoencoders (DSAE) are used for automatic feature extraction and then Softmax classifier is then used for classification purpose.

The proposed architecture is first trained on unlabelled hand images. Before training, images are first down-sampled and then are converted to greyscale. After training, the weight (transformation) matrices learned are used to transform the data. This transformed data is used as features for training the Softmax classifier.

For testing, the raw test data is transformed by stacked DSAEs and then fed into Softmax classifier for classification result. Figure II explains the above mentioned architecture.

A. Autoencoders

According to Baldi "Autoencoders are simple learning circuits which aim to transform inputs into outputs with the least possible amount of distortion" [15]. It is just a neural network with one hidden layer, which applies Backpropagation algorithm, setting target values equal to the inputs. See figure 2. As there are no class labels involved, it is an unsupervised learning technique. It tries to learn a function $h_{W,b}(x) \approx x$, where (W, b) are the network parameters known as weight and bias. It tries to learn an approximation to identity function, but if additional constraints are placed on the network such as limiting the number of hidden units to be less than the number of input units, then it learns a compressed version of the input. Also by forcing the hidden units to have mostly zero activations/values interesting representations can be learned (Sparse Autoencoders) [16].

The cost function of Autoencoder used in this paper is given by

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2 + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ij}^{(l)}\right)^2 \quad (1)$$

where,

$x^{(i)}$ is the i^{th} input vector

$y^{(i)} = x^{(i)}$

n_l is number of layers

$h_{W,b}(x^{(i)})$ is the hypothesis or observed output when input is $x^{(i)}$

λ is weight decay(regularization)parameter used to prevent overfitting. It prevents the function to overfit the data by penalizing the cost function if the value of weights increase.

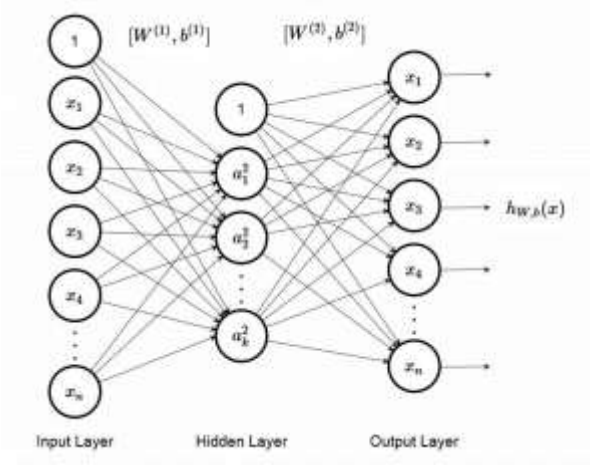

Fig. 2. Autoencoder

B. Denoising Autoencoders

Another constraint can be added to the autoencoder even if the number of hidden neurons is greater than or equal to the number of input neurons and still it will be able to find interesting patterns in the data. This constraint is to use corrupted data as input but use uncorrupted data as the output. These type of Autoencoders are called Denoising Autoencoders [17]. The input feature vector x is corrupted by randomly selecting some of the input values and setting it to zero. The corrupted input vector is denoted by $\hat{x}$. The Denoising Autoencoder takes the corrupted noisy data and try to de-noise it. See figure 3, hence it is called Denoising Autoencoder. This algorithm is motivated from the perspective of manifold learning [17].

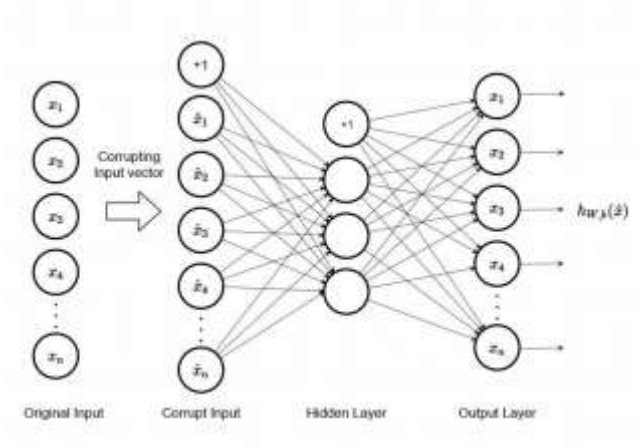

Fig. 3. Denoising Autoencoder

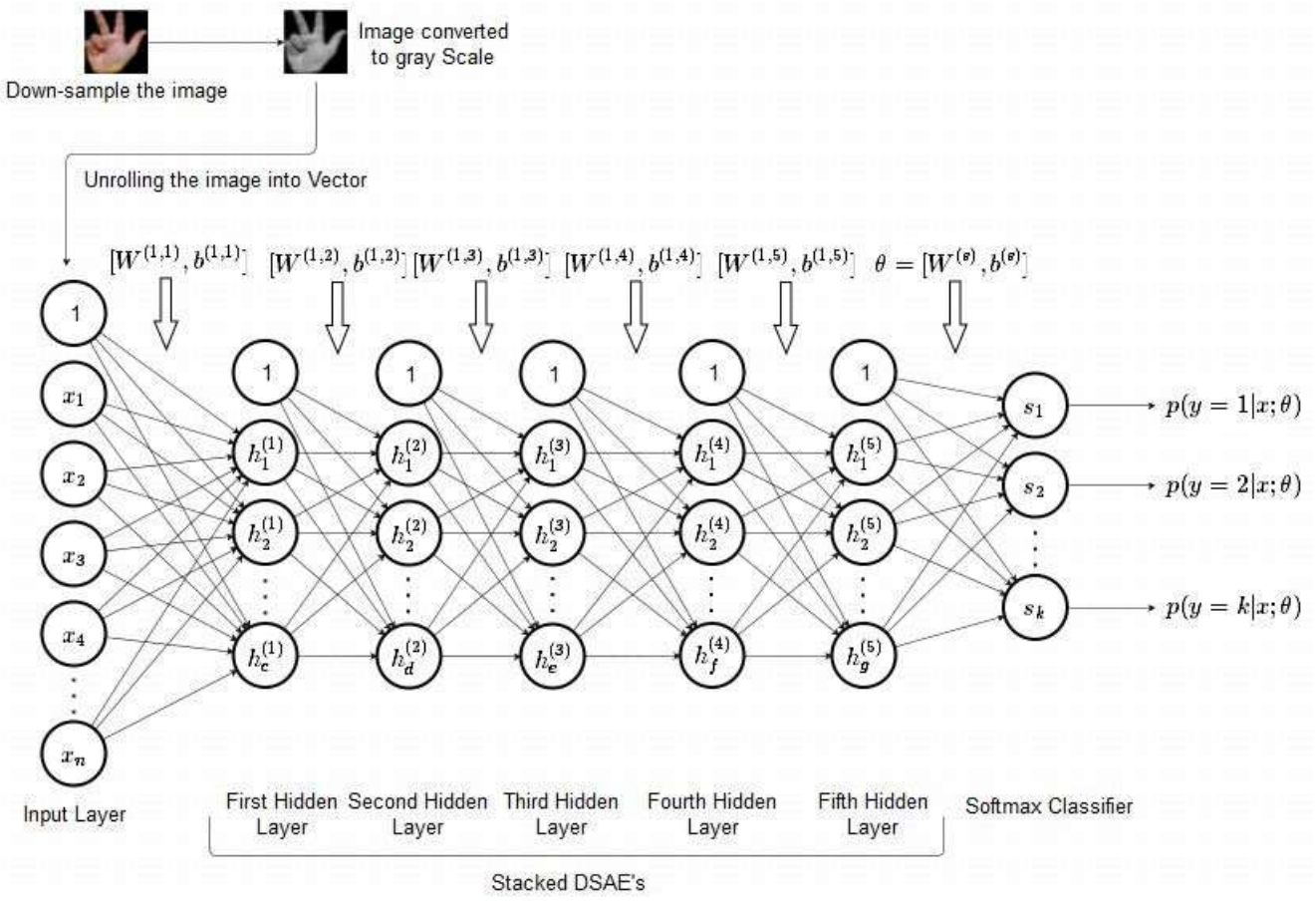

Fig. 1. Proposed architecture consisting of 5 stacked DSAE's and Softmax layer

C. Sparse Autoencoders

As discussed in section II-A additional constraints are imposed on the Autoencoder to make it learn better representation of data. In Sparse Autoencoder, sparsity constraint is added on the hidden units. Hidden units are constrained to be zero most of the time when the activation function taken is sigmoid function. This is motivated by the structure of brain in which most of the neurons are 'inactive' most of the time. To implement this constraint an additional parameter called *sparsity parameter* is used which is denoted by ρ. After every epoch we calculate $\hat{\rho_j}$ which is the average activation of j^{th} hidden neuron over the entire dataset. We penalize the cost function when $\hat{\rho_j}$ deviates from ρ. This is achieved by adding another penalty term to the cost function of autoencoder. The penalty term used in this paper is based on KL Divergence and is given as

$$\sum_{j=1}^{s_2} KL(\rho||\hat{\rho^j}) = \sum_{j=1}^{s_2} \rho \log \frac{\rho}{\hat{\rho_j}} + (1 - \rho) \log \frac{1-\rho}{1-\hat{\rho_j}} \quad (2)$$

This penalty term has a property that $KL(\rho||\hat{\rho_j}) = 0$ if $\hat{\rho_j} = \rho$, otherwise it increases monotonically as $\hat{\rho_j}$ diverges from ρ.

Now, the overall cost function of Sparse autoencoder becomes

$$J_{sparse}(W, b) = J(W, b) + \beta \sum_{j=1} KL(\rho||\hat{\rho_j}) \quad (3)$$

D. Denoising Sparse Autoencoder

Combining the properties of both the denoising and sparse autoencoder we get Denoising Sparse Autoencoder. Its objective function is same as that of Sparse Autoencoder. The only difference is that we have to feed the corrupted input into the input layer as discussed in section II-B.

E. Softmax layer

Softmax layer [18] also known as multinomial regression layer does the actual work of classification. It is a supervised learning algorithm. It is the generalized version of the logistic classifier which extends the classification to more than two classes. See figure 4. Suppose the training dataset is $\{x^{(1)}, y^{(1)}, x^{(2)}, y^{(2)}, \ldots, x^{(m)}, y^{(m)}\}$. Where $x^{(i)} \in \mathbb{R}^n$ is the i^{th} input sample and $y^{(i)}$ is the corresponding label which can take values from 1 to k, i.e. $y^{(i)} \in \{1, \ldots, k\}$ where k is the number of classes. And m is the number of

data samples. The hypothesis and the cost function are defined in section II-E1 and II-E2 respectively.

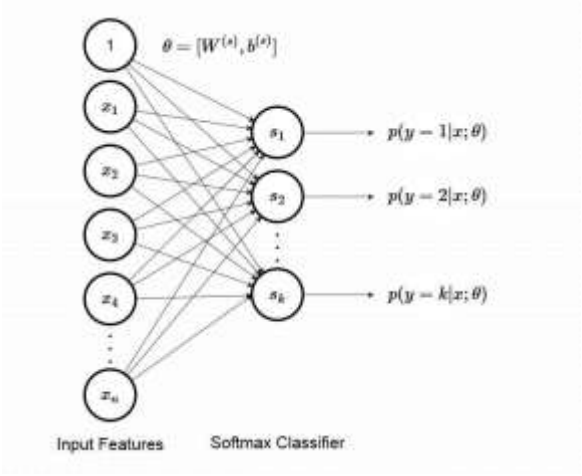

Fig. 4. Softmax Classifier

1) Hypothesis: For each test data input $x^{(i)}$, the hypothesis gives the probability $p(y^{(i)} = j|x^{(i)})$ i.e with what probability is the label $y^{(i)}$ of data sample $x^{(i)}$ belong to each of the class of j where $j = \{1, 2, 3, \ldots, k\}$.

Hypothesis is given by:

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1|x^{(i)}, \theta) \\ p(y^{(i)} = 2|x^{(i)}, \theta) \\ \vdots \\ p(y^{(i)} = k|x^{(i)}, \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix} \quad (4)$$

Where, $\theta = \begin{bmatrix} \theta_1^T \\ \theta_2^T \\ \vdots \\ \theta_k^T \end{bmatrix}$ and θ_j is the bias and weights connected to j^{th} node of the softmax classifier.

2) Cost Function: Cost function for the Softmax Classifier [18] is given as,

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} I\{y^{(i)} = j\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{s=1}^{k} e^{\theta_s^T x^{(i)}}} \quad (5)$$

Where, $I\{\cdot\}$ is called Indicator Function,

$$I\{x\} = \begin{cases} 1 \text{ if } x \text{ is True} \\ 0 \text{ if } x \text{ is False} \end{cases} \quad (6)$$

eg.

$I\{3 + 8 = 11\} = 1$
$I\{5 - 3 = 1\} = 0$

As there is no analytical solution to minimize the above cost function $J(\theta)$, iterative algorithm is applied to optimize the function. L-BFGS algorithm [19] is used in this paper.

III. Stacked Autoencoders

Autoencoders (Sparse, Denoising etc.) are used to extract meaningful features from data. The hidden layer activations $\{a_1, a_2, \ldots, a_c\}$ shown in figure 5 are actually the features extracted by the Autoencoder.

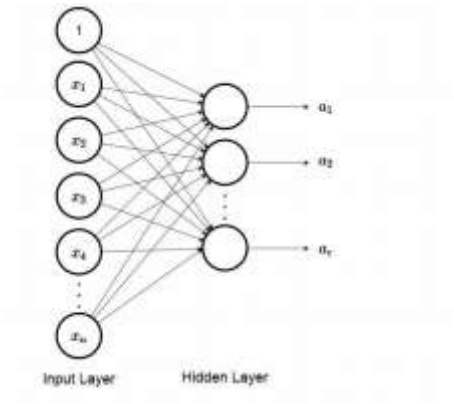

Fig. 5. Activations of hidden layer of Autoencoder

It does a non-linear operation to produce a non-linear transformation of the data. Doing another transformation on the transformed data results a more complex representation. This can be done by stacking Autoencoders on top of each other, i.e training the next Autoencoder using the hidden layer activations of the previous Autoencoder. See figure 6. Similarly any number of Autoencoders can be stacked. The number of Autoencoders stacked gives the depth of the Neural network. Traditionally Neural networks, the weights are randomly initialized. Now the weights learned by the Autoencoder are used as the weights of traditional Neural network. The notations for weights and bias for k^{th} stacked Autoencoder is given as follows.

- $W^{(1,k)}$ denotes the weight between input layer and hidden layer.
- $b^{(1,k)}$ denotes the bias between input layer and hidden layer.
- $W^{(2,k)}$ denotes the weight between hidden layer and output layer.
- $b^{(2,k)}$ denotes the bias between hidden layer and output layer.

The connections between Input layer and first hidden layer are given the weights of first Autoencoder $W^{(1,1)}$ and the connections between the first hidden layer and second hidden layer are given the weights learned by the second Autoencoder $W^{(1,2)}$ which is stacked on top the first Autoencoder. See figure 7 for more details. In this way Deep Neural networks are assign weights. For classification purposes we do not need

the weights $W^{(2,k)}$ and bias $b^{(2,k)}$. Assigning weights in this way and not randomly gives a kind of 'prior' to the network to start with. In this way it helps to avoid local optima. After the construction of the network, we add Softmax classifier in the end. This will actually classify the data into multiple classes. The previously made layers are for the purpose of feature extraction.

Therefore, now, for testing the raw data is fed into the network. Every hidden layer transforms the data till the last hidden layer. Then the final set of extracted features are then fed into Softmax classifier to be classified into one of the classes.

After constructing the full network comes the task of optimizing its cost function. In this paper L-BFGS algorithm [19] is used to optimize the cost function.

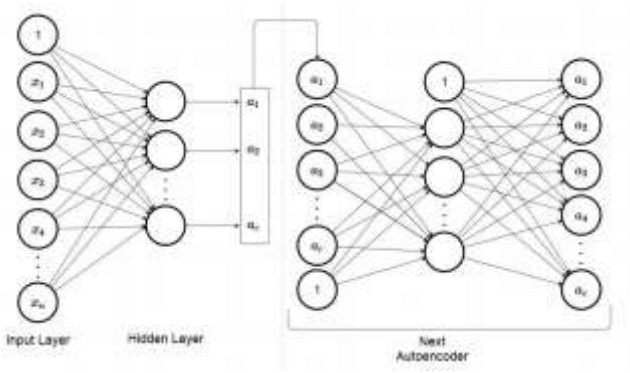

Fig. 6. Figure shows how the activations of previous Autoencoder is used to train the next Autoencoder

IV. Results

All the training and testing was done on a standard dataset [1] containing RGB images of static hand gestures. It had a total of 2425 images from 5 individuals spanned over 36 gestures of ASL of which only 20 gestures were taken. The images were converted to greyscale and resized to 28×28 pixels. The input vector was created by unrolling the image matrix into a vector of size 784×1. Originally there were only 70 samples per gesture.

Dataset was extended by adding rotations and noise to the images. After extending 700 samples per class were used for training and 420 for testing. 60,000 unlabeled images were used for training the stacked Denoising Sparse Autoencoder network. Then the features of 700 labeled training samples were extracted using the stacked Denoising Sparse Autoencoder network with each hidden layer having 200 units and then the extracted features were used to train the Softmax layer. The results are computed by taking different number of layers of DSAE with Softmax layer and finetuning the whole network using labeled data by applying Backpropagation algorithm for computing gradients and L-BFGS for optimization. Results are shown in table I.

The first hidden layer is visualized by taking the weights between input layer and each of the first hidden layer's

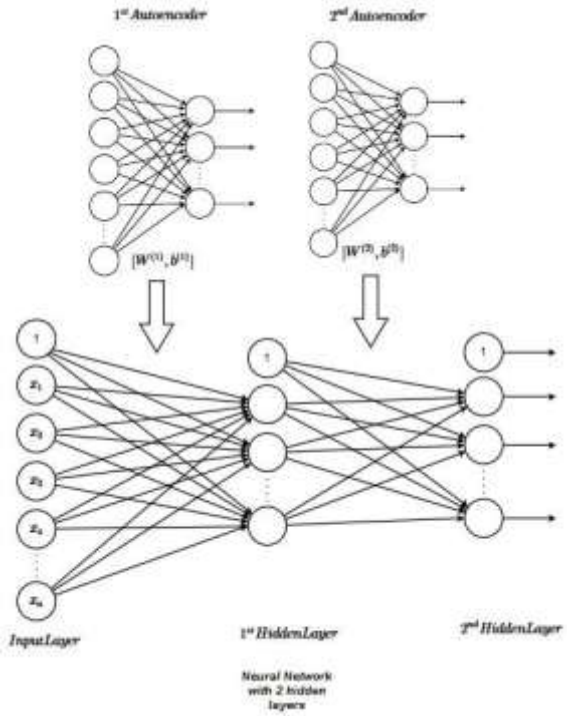

Fig. 7. Figure shows how the weights of the Neural network are replaced by Autoencoder weights

neurons, reshaping the weight vector into 28×28 matrix and displaying it as an image. Therefore each image correspond to one hidden neuron of first hidden layer. Each image shows what input will cause the corresponding hidden neuron to be 'active' or 'firing'. It is visualized in figure 8.

Another approach was to manually extract features by ap-

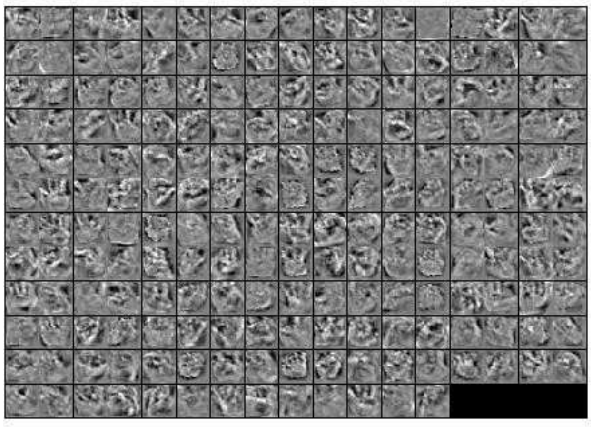

Fig. 8. **First Hidden layer visualized**

plying Principle Component Analysis. After extracting the features, they are used to train a Neural Network with one hidden layer. This result is also displayed in table I

TABLE I
CLASSIFICATION RESULTS OF DIFFERENT GESTURE RECOGNITION APPROACHES

Method for feature extraction + Cllassifier	Accuracy
DSAE 2 layers + Softmax	80.00%
DSAE 3 layers + Softmax	81.10%
DSAE 4 layers + Softmax	82.23%
DSAE 5 layers + Softmax	83.36%
PCA + Neural Network with one hidden layer	78.90%

So, it was observed that automatic feature engineering was able to give better features and hence better classification accuracy.

V. CONCLUSION AND FUTUREWORK

This paper presented Stacked Denoising Sparse Autoencoders for the use of feature extraction from images of static hand gestures and Softmax layer for the purpose of classification. Increasing the depth of Autoencoder network to 5 layers gives the classification accuracy of 83.36% . The technique used in this paper is implemented using 20 gestures, it can be extended by taking all the 36 hand gestures of American Sign Language (ASL).

Future work will consist of developing a complete human computer interaction system starting with acquiring gesture images in real time, classifying them and then performing user defined tasks. Various other invariants of autoencoders such as Contractive Autoencoders can also be tested. The computation was done in a serial fashion, use of GPU is encouraged as it will reduce the training time significantly due to parallel processing.

REFERENCES

[1] ALC Barczak, NH Reyes, M Abastillas, A Piccio, and T Susnjak. A new 2d static hand gesture colour image dataset for asl gestures. 2011.

[2] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311–324, 2007.

[3] Sören Lenman, Lars Bretzner, and Björn Thuresson. Using marking menus to develop command sets for computer vision based hand gesture interfaces. In *Proceedings of the second Nordic conference on Human-computer interaction*, pages 239–242. ACM, 2002.

[4] Rajeev Sharma, Thomas S Huang, Vladimir I PavloviC, Yunxin Zhao, Zion Lo, Stephen Chu, and K Schul. Speech/gesture interface to a visual computing environment for molecular biologists. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 964–968. IEEE, 1996.

[5] Jochen Triesch, Jan Wieghardt, Eric Maël, and Christoph Von Der Malsburg. Towards imitation learning of grasping movements by an autonomous robot. In *Gesture-Based Communication in Human-Computer Interaction*, pages 73–84. Springer, 1999.

[6] Maria Karam. *PhD Thesis: A framework for research and design of gesture-based human-computer interactions*. PhD thesis, University of Southampton, 2006.

[7] Siddharth S Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, pages 1–54, 2012.

[8] Haruhisa Kawasaki and Tetsuya Mouri. Design and control of five-fingered haptic interface opposite to human hand. *Robotics, IEEE Transactions on*, 23(5):909–918, 2007.

[9] Jochen Triesch and Christoph Von Der Malsburg. Robust classification of hand postures against complex backgrounds. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 170–170. IEEE Computer Society, 1996.

[10] Eng-Jon Ong and Richard Bowden. A boosted classifier tree for hand shape detection. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 889–894. IEEE, 2004.

[11] Peter RG Harding and Tim J Ellis. Recognizing hand gesture using fourier descriptors. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 286–289. IEEE, 2004.

[12] Pujan Ziaie and Alois Knoll. An invariant-based approach to static hand-gesture recognition.

[13] Contours : More functions. http://docs.opencv.org/trunk/doc/py tutorials/py imgproc/py contours/py contours more functions/py contours more functions.html. Accessed: 2014-06-15.

[14] S Naidoo, CW Omlin, and M Glaser. Vision-based static hand gesture recognition using support vector machines. *University of Western Cape, Bellville*, 1998.

[15] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *ICML Unsupervised and Transfer Learning*, pages 37–50, 2012.

[16] Andrew Ng. Sparse autoencoder.

[17] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[18] Andrew Ng. Softmax regression. http://ufldl.stanford.edu/wiki/index.php/Softmax Regression. Accessed: 2014-05-25.

[19] Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc V Le, and Andrew Y Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272, 2011.