Denoising and Stacked Autoencoders

Soundarya Sankarasubramanian
1213119620
ssanka24@asu.edu

Amulya Mysore
1212263102
amysore2@asu.edu

Jayashree Subramanian
1214347796
jsubram5@asu.edu

Mayuri Kambli
1212890144
mkambli@asu.edu

I. Introduction

An autoencoder is an unsupervised neural network where the output learned is the same as the input. It serves an excellent feature extractor. In an autoencoder, the input data is encoded into a code representation and the code is decoded to get the output. By varying the number of neurons in the hidden layers, interesting features of the images can be learned. There are several variations to autoencoders such as Denoising autoencoder, Sparse autoencoder, Stacked autoencoders, Variational autoencoders and so on. We have implemented denoising autoencoder with a single hidden layer. Images which were corrupted with Gaussian noise with variance v and mean 0 were trained using an autoencoder. The model was then used to analyze the denoising capabilities of the network. The network was able to handle noise up to 'v' very well and it wasn't able to reproduce the images when the image was corrupted with noise more than 'v'. Stacked autoencoders were built with three hidden layers and it gave the best accuracy of 72.8 when 10 samples per class were used for training compared to several other machine learning classifiers such as Logistic Regression, Gaussian Naive Bayes, Decision Trees, Random Forests, KNN, SVM, Bernoulli Naive Bayes, Multinomial Naive Bayes. Logistic regression gave the second best accuracy of 71.42 with lbfgs solver and random forests gave the third best accuracy with 65.06. Similarly, when just a single sample per class was used for training, stacked autoencoders performed exceptionally well with an accuracy of 53.33 which was then followed by Bernoulli Naive Bayes which gave an accuracy of 52.

A. Denoising Autoencoder

In a denoising autoencoder, some amount of noise is introduced in the input and the aim is to learn the original undistorted input. A stochastic mapping of the input which leads to a corrupted image is passed to the autoencoder. The amount of noise could be varied in each try to determine how well the network tries to obtain unique features. If x is the input passed to the first layer of the denoising autoencoder which is essentially the code representation which has to be decoded, it can be represented as $y = f_\theta(x) = s(wx + b)$ with θ consisting of $\{$w,b$\}$. From the code representation, the decoder produces $z = g_\theta(y) = s(w'y + b')$ with θ' consisting of $\{$w',b'$\}$. For w', we use w^T. When weights are represented in this way, they are called tied weights. [6]

1) Types of noises:

a) Masking Noise: If 'k' is the percentage of noise that has to be introduced, kd values will be replaced with zero where d is the number of dimensions. This kind of noise is called a masking noise. Autoencoder will try to fill in all these blanks by being robust to noise.

b) Gaussian Noise: Noises added to the image can be gaussian with mean and variance. We have implemented denoising autoencoders with gaussian noise having mean 0.

c) Translations: The input image can be rotated or some pixels can be set to 0. Some pixels of the image can be replaced with pixels from other images.

B. Stacked Autoencoder

Stacked Autoencoder consists of several layers of sparse autoencoders. The output of each layer is connected to the input of the next layer. Stacked autoencoders perform layer wise training. Good parameters can be obtained in a greedy fashion. During training, the first layer is trained on the input to obtained the parameters $W^{(1,1)}$,$W^{(1,2)}$,$b^{(1,1)}$,$b^{(1,2)}$. The output of this layer will yield a vector and this vector is used to train the second layer and the parameters $W^{(2,1)}$,$W^{(2,2)}$,$b^{(2,1)}$,$b^{(2,2)}$ are learnt. [5] The first order features such as edges can be learnt when the input is passed to the first layer. The second layer learns special patterns in terms of the edges detected. The higher layer captures the higher order features. Stacked autoencoder has the encoding and the decoding steps. The encoder step follows the forward propagation and decoder is implemented in the reverse order.

$$a^{(l)} = f(z^{(l)})$$
$$z^{(l+1)} = w^{(l,1)}a^{(l)}+b^{(l,1)}$$
$$a^{(n+l)} = f(z^{(n+l)})$$
$$z^{(n+l+1)} = w^{(n-l,2)}a^{(n+l)}+b^{(n-l,2)}$$

Once the stacked autoencoder is built, fine tuning is performed to improve the accuracy of the network. Fine tuning involves freezing the learning rate for the stacked layers or giving a very low value to the learning rate for the stacked layers. [5]

II. Related Work

The Fashion-MNIST dataset has been tested for different classifiers for its performance and it stands as a benchmark for testing machine learning algorithms [15]. An alternative to denoising autoencoders was proposed by Doi.et.al in which the aim was to obtain uncorrupted input by passing the input without noise through a linear encoder which is corrupted. Translations were performed to training data and passed along

with the actual training data to the network and doing this improved the performance significantly. The relationship between training the inputs with noise and applying regularization to the network was witnessed by Bishop. But denoising autoencoders performed a lot better than a regularized autoencoder [6]. In order to introduce noise to the images, the images were translated by rotation, some patches of the image were replaced with patches from other images. Some percentage of the random pixels were set to zero. For each of these, the performance of the Stacked denoising autoencoders was observed. As the noise level was increased for the denoising autoencoders, more discriminating features were learned by the network. The deep denoising autoencoders have been implemented for speech enhancement [17]. The denoising autoencoder learns to discriminate between the speech and noise, which leads to the removal of noise to improve speech quality. The deeper the network will give better speech quality. Stacked autoencoders have been used for multiple organ detection from the unsupervised 4D DCE-MRI medical dataset [7]. The stacked autoencoders have been used in land-cover mapping to classify large-scale remote-sensing images. A stacked autoencoder model was implemented for African land-cover mapping [10]. When compared with Random Forest, SVM and ANN, the stacked autoencoder performed better with an accuracy of 73.99%. In the domain of security stacked autoencoders have been used for detecting application level DDoS attack by learning the useful features of the network [11]. Applications including effective insect recognition [12], classification and diagnosis of Parkinson's disease [13], human activity recognition [14] used stacked autoencoders to obtain high performance and were good feature extractors for the classification in comparison with other baseline leaning models. A study shows that when these both autoencoders are combined, stacked denoising autoencoder can detect the Gabor-like edges in natural image patches and larger strokes in digit images [8]. Thus, stacked denoising autoencoders performed much better than the stacked basic autoencoders as seen from the previous work [6]. The stacked denoising autoencoders have also been used to generate bottleneck features in speech recognition by training deep neural networks by layer-wise in unsupervised manner [9]. Other data-oriented approaches like generalized autoencoders have been proposed which can efficiently handle learning for more complex datasets [16].

III. Method

MNIST Fashion dataset consists of a training set of 60,000 examples and a test set of 10,000 examples. MNIST fashion dataset includes 10 labels and they are T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot. MNIST fashion dataset was built because the digits dataset was assumed to be very easy for several machine learning algorithms.

A. Denoising Autoencoders

Denoising autoencoder is built by passing the corrupted version of the images as input. This corrupted version is obtained through some stochastic mapping of the input. The idea is to build a network which is robust to noise. We implemented a denoising autoencoder with a single hidden layer. Gaussian noise with mean 0 and variance v was added to the image. Different values of v experimented include 0.1, 0.5,1,5,10,20. The image added with noise was passed as input to the autoencoder. The aim of the denoising autoencoder is to construct denoised images from the corrupted input. The images were initially trained by adding Gaussian noise with mean 0 and variance 0.5 as the input. The network was trained by updating the parameters according to the loss function used. Reconstruction loss was used to compute the error. The error was computed with the output of the autoencoder and the actual image instead of the corrupted image. 59000 images of the MNIST Fashion dataset were used for training and the type of gradient descent used was Batch. The performance of the network can be evaluated on the basis of how robust it is with respect to noise.

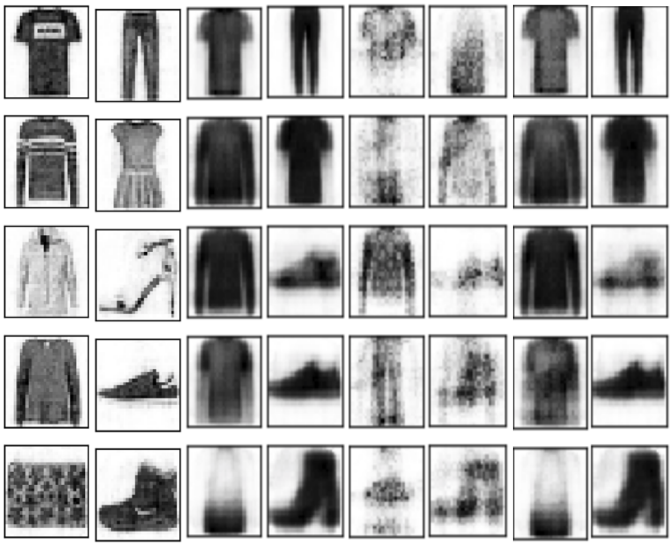

Fig. 1. Denosing autoencoders a) Training with Gaussian noise mean 0, variance 0.5 (column 1,2) b) Test Image with Gaussian noise mean 0, variance 0.1 (column 3,4) c) Test Image with Gaussian noise mean 0, variance 10 (column 5,6) d) Test Image with Gaussian noise mean 0, variance 0.5 (column 7,8)

The relationship between the percentage of noise introduced to the input and the amount of loss incurred was plotted to see how the loss changes with respect to the noise. Generally, for autoencoders, the number of neurons in the hidden layers are larger when compared to the input layer. This is done with the belief that the network will not learn the identity function and instead try to learn some useful features of the input. The advantage of denoising is that since the input is corrupted, even if the number of neurons in the hidden layer is equal to the number of neurons in the input layer, if the output obtained is an uncorrupted version of the input, we can confirm that the network learns the useful features and does not perform a regular identity function. With this, we can even have hidden layers larger than the input layer. In practice, the majority of

denoising autoencoders are built with largely hidden layers. In our implementation, a hidden layer with different sizes of neurons such as 900, 1000, 1100, 1200, 1500 and 2000 was tried. There was not much difference in the cost values when we varied the number of neurons in the hidden layer. With the increase in the number of neurons in the hidden layer, there was an increase in the duration of the computations.

B. Stacked Autoencoders

Autoencoders are excellent unsupervised feature extractors. Three autoencoders were built in an unsupervised manner with dimensions 784-500-784, 500-200-500 and 200-100-200. The architecture of the stacked autoencoder has 784 neurons in the input layer followed by 500 neurons in the first hidden layer, 200 in the second hidden layer and 100 in the third hidden layer. These autoencoders were trained separately for a finite number of iterations. The encoders of these layers were then stacked together and a layer was added at the top for classification. The stacked autoencoder is then fine-tuned with labeled samples. The fine tuning is performed by freezing the stacked layers by giving a learning rate 0 or very less learning rates like 0.001, 0.0001. The learning rates and the number of iterations were varied in order to find the optimal parameters for which the cost was minimal. Batch gradient descent was used and it enabled us to test whether our learning rates were working to reduce the cost in each step. Reconstruction loss was calculated using the Euclidean distance which was then used by the autoencoder to learn and update the weights. These stacked autoencoders were then compared with the baseline models to gauge its performance and efficiency.

IV. Experiments

A. Denoising Autoencoders

Weights for the neural network were initialized with random normal values and they were multiplied by a factor of sqrt(1/Number of neurons in the output layer). The number of neurons in the hidden layer of the denoising autoencoder was varied by several other choices such as 900, 1000, 1100, 1200, 1500, 2000 and so on. The table below contains the testing costs when the images corrupted with gaussian noise having mean 0 and variance 0.1 were used for training.

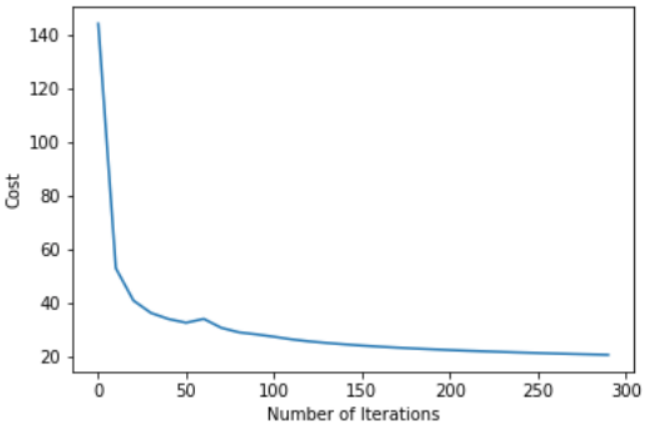

Fig. 2. Denosing autoencoders Cost Vs. Number of iterations

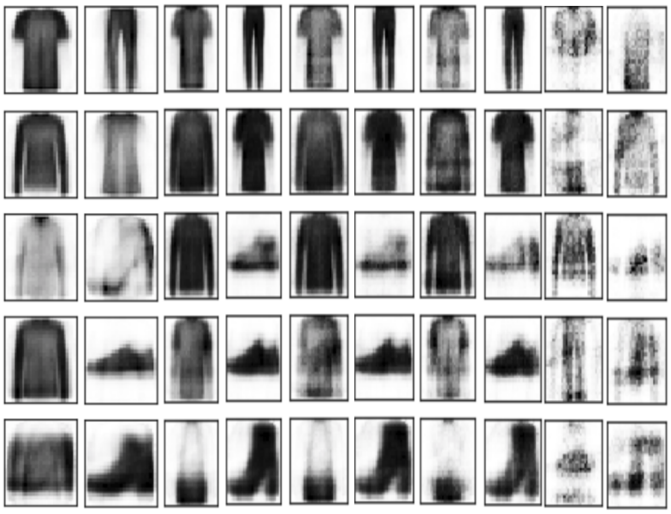

Fig. 3. Denoising autoencoders a) Training with Gaussian noise mean 0, variance 0.1 (column 1,2) b) Testing with Gaussian noise mean 0, variance 0.1 (column 3,4) c) Testing with Gaussian noise mean 0, variance 0.5 (column 5,6) d) Testing with Gaussian noise mean 0, variance 1 (column 7,8) e) Testing with Gaussian noise mean 0, variance 5 (column 9,10)

Testing with Gaussian Noise	Cost
Variance 0.1	16.48
Variance 0.5	18.19
Variance 1.0	24.14
Variance 5.0	88.32

The best cost was obtained with 1200 number of neurons in the hidden layer. But in order to train faster with 60,000 samples, we trained using 1000 neurons in the hidden layer. We trained the network with Gaussian noise with variance of 0.5 and observed the cost values of the network with the increase in the number of iterations. Since we used batch, the cost was supposed to reduce at each step and hence we could confirm that our learning rates were working properly. We used the model to test for images added with noise containing 0.1, 0.5,1,5 variances. We measured the cost values of these images for testing and we could find that when the model was trained with a variance of 0.1, it could handle the images added with 0.1, 0.5 noise decently well but it didn't work for higher values of noise. Similarly, when the images were corrupted with Gaussian noise with variance of 0.5, during testing, the network was able to handle images added with Gaussian noise of 0.1 and 0.5 variances and it wasn't able to produce any clear image for higher values of variance.

B. Stacked Autoencoders

Weights for the three autoencoders were initialized with random normal values and they were multiplied by a factor of sqrt(1/Number of neurons in the output layer). While building

the three autoencoders, the number of neurons were varied several times to get the one that works very well. Alternatives such as 784 - 500 - 200 - 100 - 10, 785 - 500 - 100 - 50 - 10, 784 - 600 - 200 - 100 - 10 were tried. There was not much difference in the performances of these different architectures. However, we used the architecture with dimensions 784 - 500 - 200 - 100 - 10 for classification. Since autoencoders are unsupervised, we used 10000 samples to train the three autoencoders separately. The features extracted were used for the stacked autoencoder. The last layer was initialized with random normal values multiplied by a factor of sqrt(1/Number of neurons in the output layer). Fine tuning was performed by freezing the stacked layers and by only updating the weights of the final layer. The test accuracy observed was 72.8 and it was found to be better than the accuracy of all the other models when training was done with 5 samples per label. The accuracies for all the classifiers when 5 samples per class were used for training is shown in the table below.

Dimensions	Learning rate	Number of iterations	Cost
$784-500-784$	0.6	1500	0.137
$500-200-500$	0.5	500	0.095
$200-100-200$	0.5	500	0.113

Iterations	Learning rate except last layer	Decay rate	Learning rate Last layer	Test accuracy
1500	0	0.01	2	72.80
1500	0.001	0.01	2	72.70
1500	0.0001	0.01	2	72.70

Similarly, when the 1 sample per label was used for training, stacked autoencoders again gave the best accuracy of 53.33. The accuracies for all the classifiers when 1 sample per class was used for training is shown in the table below.

Dimensions	Learning rate	Number of iterations	Cost
$784-500-784$	0.6	1500	0.12
$500-200-500$	0.5	500	0.03
$200-100-200$	0.5	500	0.002

Iterations	Learning rate except last layer	Decay rate	Learning rate Last layer	Test accuracy
1000	0	0.01	0.5	53.33
1000	0.001	0.01	0.5	52.31
1000	0.0001	0.01	0.5	52.31

V. Conclusion

Denoising autoencoders were able to reconstruct the original images when Gaussian noise was added to the images. The reconstructed images obtained as a result of denoising had little loss. However, when extremely noisy images were used to test, it didn't perform well as expected. Autoencoders are capable of learning very good features from the data in an unsupervised fashion. Stacked autoencoders gave an excellent accuracy of 72.80 when 5 samples per label were used for training and they performed better than most of the classifiers such as Logistic Regression, Random forests, Multinomial Naive Bayes, SVM, Bernoulli Naive Bayes, Decision Trees, and Gaussian Naive Bayes. Accuracies for various machine learning algorithms are recorded in the decreasing order when one sample per class i.e a total of 10 samples were used.

Classifier Algorithms	Test accuracy
Stacked Autoencoder	53.33
Bernoulli Naive Bayes	52.89
Multinomial Naive Bayes	49.98
Logistic Regression (lbfgs solver)	47.86
Logistic Regression (newton cg)	47.86
SVM	45.76
Gaussian naive bayes	45.76
Random forests	37.00
Decision Trees	30.16

With just one sample per class, Stacked autoencoders gave an accuracy of 53.33 when compared to other classifiers like Bernoulli Naive Bayes, Logistic regression, SVM, Random forests, Decision Trees, Gaussian Naive Bayes. Accuracies for various algorithms are recorded in the table below when 5 samples per class, i.e for a total of 50 samples were for training. The accuracies are listed in the decreasing order in the table.

Classifier Algorithms	Test accuracy
Stacked Autoencoder	72.80
Logistic Regression (lbfgs solver)	71.42
Logistic Regression (newton cg)	71.35
Random forests	65.06
Multinomial naive bayes	64.6
SVM	61.39
Bernoulli Naive Bayes	60.61
Decision Tress	53.56
Gaussian naive bayes	34.71

VI. Division of work

The work was equally divided among all the four members. Working of denoising autoencoders was done by Soundarya Sankarasubramanian and Amulya Mysore. The working and fine tuning of stacked autoencoder were done by Jayashree Subramanian and Mayuri Kambli. Baseline methods were implemented by Soundarya Sankarasubramanian.

VII. Self Peer Evaluation Table

Soundarya Sankarasubramanian	Amulya Mysore	Jayashree Subramanian	Mayuri Kambli
20	20	20	20

REFERENCES

[1] http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/

[2] A. Ng, Stacked Autoencoders, 2018 (accessed October 20, 2018). [Online]. Available: http://deeplearning.stanford.edu/wiki/index.php/Stacked Autoencoders

[3] H. Larochelle, Neural networks [6.6] : Autoencoder - denoising autoencoder, 2018 (accessed October 20, 2018). [Online]. Available: https://www.youtube.com/watch?v=t2NQ c5BFOc

[4] http://deeplearning.stanford.edu/wiki/index.php/Fine-tuning_Stacked_AEs

[5] http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

[6] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, Extractingand composing robust features with denoising autoencoders, in Pro-ceedings of the 25th international conference on Machine learning.ACM, 2008, pp. 10961103

[7] Hoo-Chang Shin, Orton, Collins, Doran, and Leach. "Stacked Autoencoders for Unsupervised Feature Learning and Multiple Organ Detection in a Pilot Study Using 4D Patient Data." Pattern Analysis and Machine Intelligence, IEEE Transactions on 35, no. 8 (2013): 1930-943

[8] Pascal Vincent,Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion." Journal of Machine Learning Research 11 (2010) 3371-3408

[9] Gehring, Jonas, Yajie Miao, Florian Metze, and Alex Waibel. "Extracting Deep Bottleneck Features Using Stacked Auto-encoders." Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, 2013, 3377-381.

[10] Weijia Li, Haohuan Fu, Le Yu, Peng Gong, Duole Feng, Congcong Li, and Nicholas Clinton. "Stacked Autoencoder-based deep learning for remote-sensing image classification: a case study of African land-cover mapping." International Journal of Remote Sensing, 2016.

[11] Yadav, Satyajit, and Selvakumar Subramanian. "Detection of Application Layer DDoS Attack by Feature Learning Using Stacked AutoEncoder." Computational Techniques in Information and Communication Technologies (ICCTICT), 2016 International Conference on, 2016, 361-66.

[12] Yu Qi, Cinar, Souza, Batista, Yueming Wang, and Principe. "Effective Insect Recognition Using a Stacked Autoencoder with Maximum Correntropy Criterion." Neural Networks (IJCNN), 2015 International Joint Conference on 2015 (2015): 1-7.

[13] H. Badem, A. Caliskan, A. Basturk and M. E. Yuksel, "Classification and diagnosis of the parkinson disease by stacked autoencoder," 2016 National Conference on Electrical, Electronics and Biomedical Engineering (ELECO), Bursa, 2016, pp. 499-502.

[14] Badem, Hasan, Abdullah Caliskan, Alper Basturk, and Mehmet Emin Yuksel. "Classification of Human Activity by Using a Stacked Autoencoder." Medical Technologies National Congress (TIPTEKNO), 2016, 2016, 1-4.

[15] Han, Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms." ArXiv.org, 2017, 15.

[16] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang. "Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction." Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, 2014, 496-503.

[17] Lu, Xugang, Yu Tsao, Shigeki Matsuda and Chiori Hori. "Speech enhancement based on deep denoising autoencoder." INTERSPEECH (2013).