Experiments with De-noising and Stacked Autoencoders

Raman Ahuja, Bhaumik Choksi, and Akshay Patil

Abstract—Feature extraction is an important preprocessing step when attempting to solve a problem using machine learning. In this report, we discuss two different types of autoencoders, which are a type of neural network that perform feature extrac- tion using deep learning. The experiments show how they can be used to extract hidden features from the input data, eliminate noise, the reduce the input size while performing classification tasks. We also analyze their performance by varying the type of inputs given, and the number of training samples provided.

Index Terms—Autoencoders, Feature extraction, Unsupervised learning, Multi-class classification

I. Introduction

FEATURE extraction plays a very decisive role in the pattern recognition tasks. The preliminary condition for classifying/reconstructing data is to find an optimal set of features. As the size of the data gets large, the amount of redundancy in it increases as well. The extraction of the most relevant information from the original input and representing this information in a lower dimension while still describing the input is a crucial step in machine learning tasks. We in this paper analyse encoder functions which learn representation of the input in lower dimension.

A. Motivation

The deep neural architectures when trained for classification with supervised objective example gradient descent, do not perform well if we start with randomly initialized parameters. They also have a more probability of getting stuck in local optima. Taking this problem as motivation, Hinton et. al. [1] proposed an approach instead of randomly initializing parameters, each layer, in turn, is pre-trained with local unsupervised criteria. This helps in learning intermediate representations from the output of each previous layer, which was observed to give more robust results in terms of the generalization of performance.

B. Autoencoders

One of the primary objective of implementing an autoencoder is to discover hidden structure, usually in the form of latent feature representation. Autoencoder consists of an encoder function which converts the input to a particular representation and a decoder function for getting the input back from the representation. Autoencoder, this way become useful when we have a hidden layer smaller in size compared to the input layer, forcing the encoder function to learn a compressed representation of the data by extracting correlations present in it. The two most intuitive applications of autoencoders are data compression and denoising. A denoising autoencoder learns the hidden representation by re-constructing data from the corrupted (by adding noise to the input) forms of the input. A further enhanced version this is the stacked autoencoder, which consists of multiple autoencoders layers where the input of a layer is the output of the preceding layer. Stacked autoencoders uses a greedy layer-wise approach for training the weights of the network. We have analysed this models for the Fashion-MNIST [2] data set below.

II. Related Work

One of the earliest works focusing on developing an algorithm to minimize the reconstruction error of training examples and higher error otherwise was proposed by Ranzato [3]. This approach viewed autoencoders as approximating an energy function through the reconstruction error and incorporated regularization which prevented learning of Null functions - Identity functions. The base idea was to not randomly initialize the deep architecture, which yielded poor results, but to instead have an unsupervised learning step at the beginning which will map the input to a robust intermediate representation. It is known that increased levels of useful architecture are crucial to obtain a higher modeling power. But when it comes to learning in the deep network model, it is observed that it is difficult. Earlier, the arduous task of initializing and optimizing of the multilayer neural network had prevented researches from experimenting with a model containing greater than 2 hidden layers. However, this issue has been solved by the work of Hinton et al. [1] for the training of Deep Neural Network and the stacked version of them. Their approach uses an unsupervised training method to perform a layer- wise initialization where a layer is trained to obtain a hidden representation of the patterns observed in the inputs depending on the input received from the previous layer. This approach used by many has been found to prevent getting stuck with poor solutions that usually result from random initialization of the layers. But, there is still no clarity on what the representation should be like for initialization of multi-layer neural networks.

Vincent et al. [4] proposed a modification to the previous approach of training deep neural networks. Their method describes how this can be achieved by a new training principle based on the idea of making the learned representations robust to partial corruption of the input pattern. This approach was used to train autoencoders, and these denoising autoencoders

were stacked [5] to initialize deep architectures instead of using a random initialization. To verify the robustness and how well the hidden layers have learned to represent the initial input features, the decoding was done at the final layer. The output has the number of same dimensions as the input. The accuracy is calculated based on the difference between the output generated and the input that was provided. This helps in identifying how well the hidden layers have learned to represent the input features. The intermediate representation can then be used to perform classification tasks. It was observed this approach of initializing the model yields much better performance for their respective tasks.

III. Methodology

A. De-noising Autoencoder

The de-noising autoencoder is trained to reconstruct a ”clean” repaired version of a corrupted input image. This can be achieved by preparing data that has been corrupted by various means. One way of doing this to corrupt an image is to set a given fraction of the pixel values in the image to zero. After corruption, we normalize the data. Once our data is prepared, the De-Noising Autoencoder is trained. This model has one hidden layer and the dimensions of the input layer and output layer are same. Here our intention is to make hidden layer learn a robust intermediate representation. There are two possible ways to represent hidden layer neurons - overcomplete hidden layers and under-complete hidden layers. With the overcomplete layer, we make the network learn a higher dimensional representation with more number of neurons than the input layer. The overcomplete representation is what we are using for our De-noising Autoencoder, as shown in Fig. 1. With the under-complete layer, we try to represent the data in low dimensional space. The sigmoid activation function used for training the network.

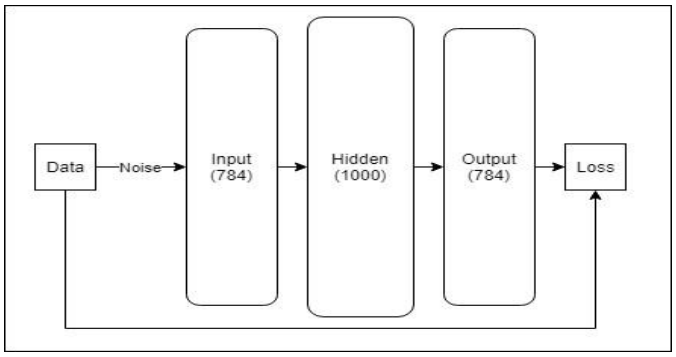

Fig. 1: Architecture of the De-Noising Autencoder

B. Stacked Autoencoder

A greedy layer-wise approach is used to train the stacked autoencoder in a layer-by-layer manner. Each layer is trained individually by freezing the parameters of the previous layers. In this approach, we use the simple autoencoder as a building block, and we train in a way that the representation learned by the i^{th} layer is used as an input to the $(i+1)^{th}$ layer. Therefore, the $(i + 1)^{th}$ layer is trained after the training of the i^{th} layer. Once a layer has been trained, we discard the decoding part of the autoencoder in that layer, and the encoded representation is fed as input to the next layer. In this manner, layer-by-layer training is performed to learn intermediate representations of decreasing sizes, without any abrupt loss in information. Once we are done with training hidden layers, we stack them layer- by-layer to form a stacked Auto-encoder.

We experimented with multiple ways of training the Stacked Autoencoder by varying the number and sizes of the intermediate layers. The standard architecture involved 3 hidden layers of sizes 500, 300 and 100 respectively, as shown in Fig. 2. Another variation involved using only one hidden layer of size 100. The 100-dimensional representation was then fed into the classifier.

C. Performing Classification

Classification is performed using the reduced intermediate representation which is generated by the autoencoder. On top of final layer of stacked autoencoder we add a dense layer and a softmax output layer for classification. This approach of classification instead of directly training a deep neural architecture with randomly initialized parameters has shown to yield significantly better generalized classification. Hinton [1] has shown in his work this approach also yields better local minima.

IV. Experiments

A. Dataset

We intend to test the given autoencoder models on Fashion-MNIST dataset [2]. This dataset consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. It is very similar to the standard MNIST dataset. For the de-noising autoencoder, we test the model with various noise levels and analyze the level of noise in the reconstructed image. For the stacked autoencoder, we test the model on subsets of the dataset. The subsets contain 10 samples (1 per class) and 50 samples (5 per class). We also vary the learning rate and analyze the impact it has on the training.

B. De-noising Autoencoder

1) Adding noise: In order to introduce noise into images, we set a user-specified percentage of pixels in the image to zero. The pixels are randomly selected from the image. We experimented with different noise levels for training and testing.

2) Training: We train the autoencoder on all the images in the training dataset with a batch size of 128 and a learning rate of 0.1. The trained autoencoder is tested on 10 randomly selected images from the test set. The training loss for training noise set to 10% can be seen in Fig. 6 and for training noise set to 20% can be seen in Fig. 7.

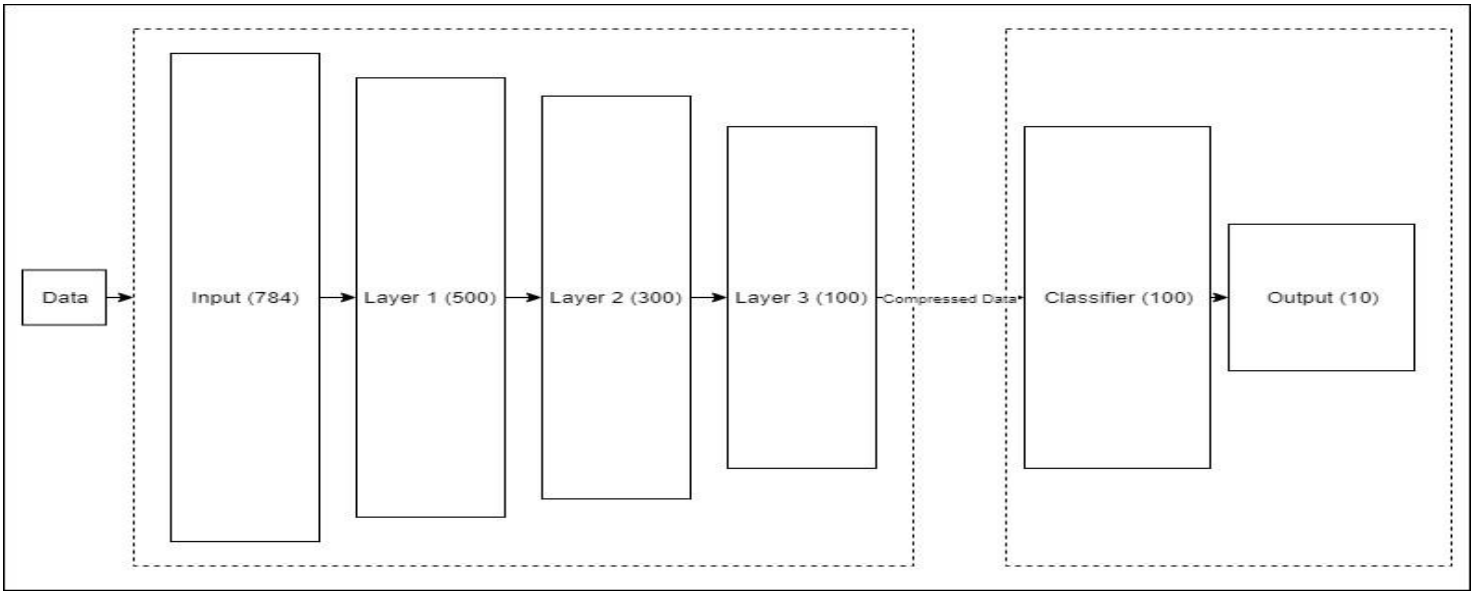

Fig. 2: Architecture of the Stacked Autencoder

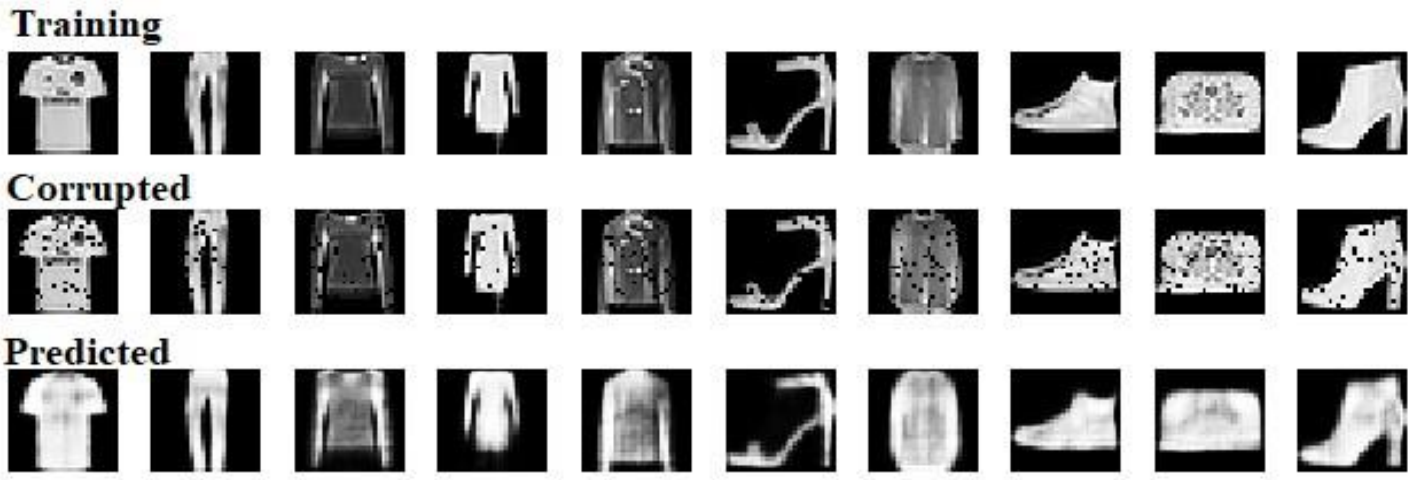

Fig. 3: De-noising autoencoder with the training noise percentage set to 10 percent and test noise set to 10 percent

3) Results: When the training noise was low, set to 10 %, the results were relatively good. As the testing noise went up, the quality of the generated images deteriorated. The results are visualized in Fig. 3, Fig. 4 and Fig. 5.

C. Stacked Autoencoder

1) Training individual layers: Each individual autoencoder has sigmoid activations on the hidden and output layers. They are trained with a batch size of 128 and a learning rate 0.1. The input for the first layer is the training data, and for each subsequent layer, the output of the previous layer acts as the input.

2) Stacking Autoencoders: Once each of the individual layers are trained, the decoding part of each of these layers is discarded, and the encoding part is stacked with the encoding part of other layers.

3) Training the classifier: The output of the stacked autoencoder is fed into a one-layer classifier, followed by an output layer of size 10. The output layer uses softmax activation and cross-entropy loss. The training accuracies for the baseline models are shown in Table. I and the accuracies for our classifier as shown in Table. II.

TABLE I: Baseline results using standard classifiers

Classifier	Training Accuracy	Test Accuracy
KNN	78.9%	75.6%
Naive Bayes	69.1%	66.5%

V. Conclusion

Experimental results show how our implementation of a simple De-noising Autoencoder with the objective of obtaining

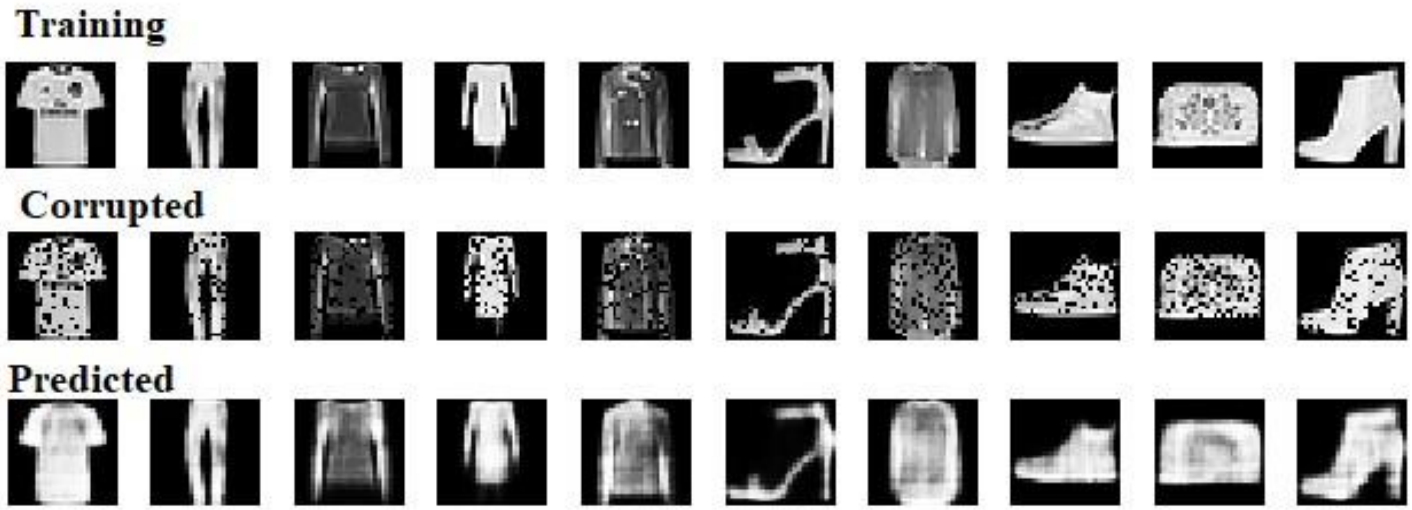

Fig. 4: De-noising autoencoder with the training noise percentage set to 10 percent and test noise set to 20 percent

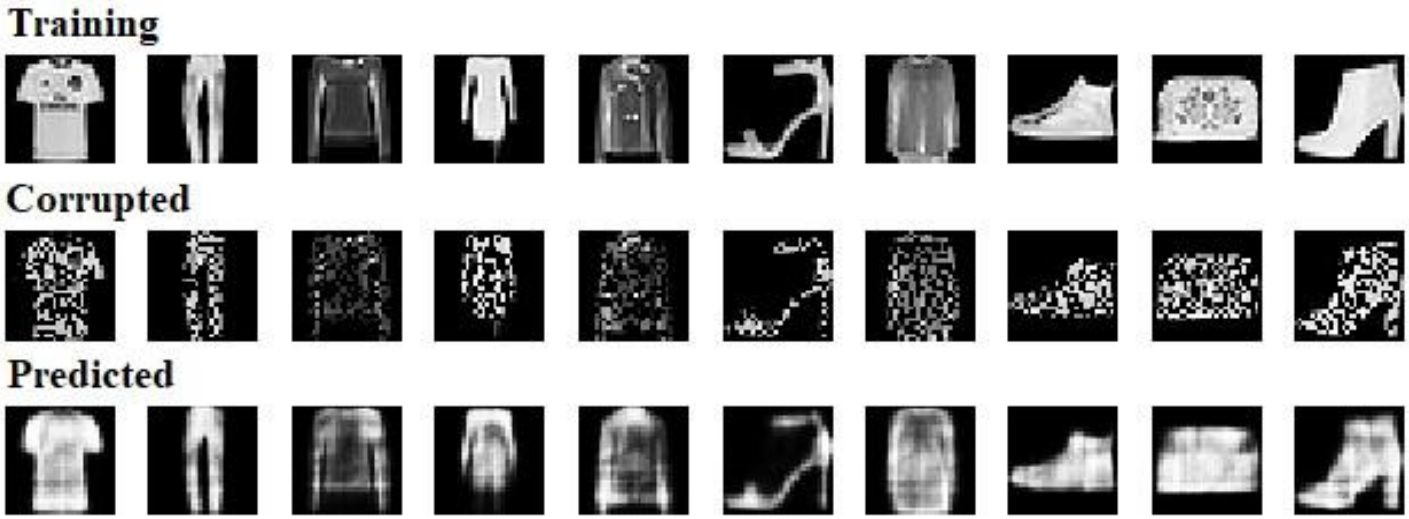

Fig. 5: De-noising autoencoder with the training noise percentage set to 20 percent and test noise set to 50 percent

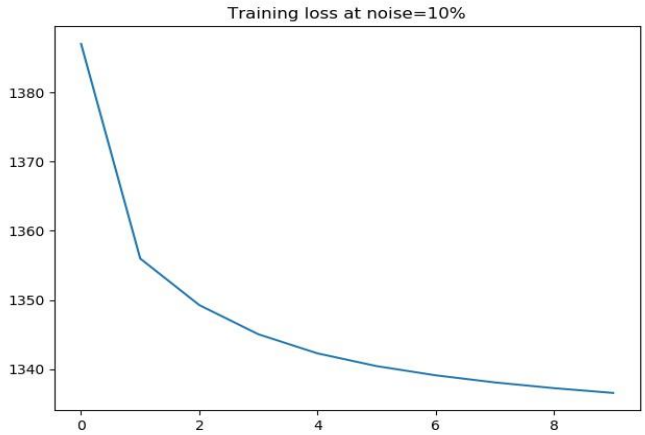

Fig. 6: Training loss at noise = 10 %

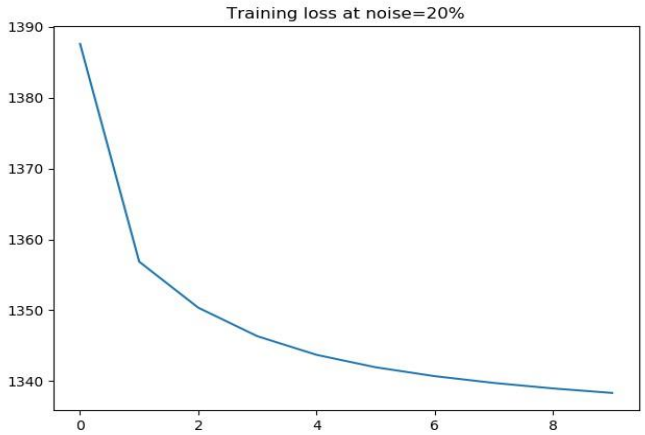

Fig. 7: Training loss at noise = 20 %

TABLE II: Results for the Stacked-Autoencoder based classifier

Architecture	Samples/class	Train Accuracy	Test Accuracy
784-500-300-100	1	60%	46.9%
784-500-300-100	5	68%	48.2%
784-100	1	40%	35.2%
784-100	5	46%	36.7%

the original input is able to undo the corruption of the input data. This was built with the motivation to learn a robust representation of the input data irrespective of the corrupted input data. Then, this principle is used to train the layers of the Stacked Autoencoders. These models were trained on the Fashion-MNIST dataset. Based on the results obtained, the model was able to learn an alternative representation of the input. These robust representations learned by the Stacked Autoencoder can then be used for supervised classification, in order to decrease the size of the input, thereby reducing the complexity of the classifier. Here, we compared our results with some baseline models such as Na¨ıve Bayes and K- Nearest Neighbors Classifiers to verify with the result the implemented models gave. The results demonstrate that an effective classifier can be trained using only a few training samples per class since the Autoencoder acts as a robust feature extractor.

APPENDIX A
GRADIENT DESCENT ALGORITHM

Gradient Descent is an iterative optimization technique that is used in various machine learning problems to minimize the cost function in the training phase. The weights of a layer, θ_j , are updated using the following rule:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

where J is the cost function and α is the learning rate.

ACKNOWLEDGMENT

We would like to thank Prof. Hemanth Venkateswara for providing insight and expertise that greatly assisted our project.

REFERENCES

[1] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh."A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.

[2] Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." arXiv preprint arXiv:1708.07747 (2017).

[3] Ranzato, Marc'Aurelio, and Martin Szummer. "Semi-supervised learning of compact document representations with deep networks." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

[4] Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

[5] Vincent, Pascal, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." Journal of machine learning research 11.Dec (2010): 3371-3408.