Dimensionality Reduction: A Comparison Between Principal Component Analysis and Deep Learning Approaches

Thesis submitted by

Fazal Noor

00000146604

Supervisor

Dr.Firdos Khan

in partial fulfilment of the requirements

for the award of the degree of

Master of Science

in

Statistics

Department of Mathematics,School Of Natural Sciences

National University Of Sciences and Technology, Islamabad

FORM TH-4

National University of Sciences & Technology

MS THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by: **Fazal Noor**, Regn No. **00000146604** Titled: **"Dimensionality Reduction: A Comparison Between Principal Component Analysis and Deep Learning Approaches"** accepted in partial fulfillment of the requirements for the award of **MS** degree.

Examination Committee Members

1. Name: DR. TAHIR MAHMOOD — Signature: [illegible]

2. Name: DR. SHAKEEL AHMED — Signature: [illegible]

Supervisor's Name: DR. FIRDOS KHAN — Signature: [illegible]

[illegible]
Head of Department

12/09/2022
Date

COUNTERSINGED

Date: 13.9.2022

[illegible]
Dean/Principal

Author's Declaration

I declare that this dissertation was written by me, and unless otherwise expressly stated in the article, the works contained in this article is my own and have not submitted for any other degree or professional qualifications. This is an exact copy of the thesis, including any final revisions required by my examiners. I understand that my thesis may be made available to the public electronically. I understand the ethical implications of my research.We have properly cited and referenced the original sources whenever we have used materials (data, theoretical analysis, and text) from other sources in the text of the report. I also declare that I have adhered to all principles of academic honesty and integrity, and that I have not misrepresented, fabricated, or falsified any idea/data/fact/source in my submission, and that I have followed the Institute norms and guidelines, as well as the regulations outlined in the Institute's Ethical Code of Conduct.

Name:	**Fazal Noor**
Enrollment Number:	**00000146604**

Abstract

The emergence of big data has increased the importance of dimensionality reduction in every field related to data analysis. The search for an optimal and generalized dimensionality reduction method is very important to reduce and optimize the time needed for data execution. Dimensionality reduction is the most needed part in any type of data pre-processing and visualization. The idea of this research study is basically to compare deep learning methods (Auto-encoders) with principal component analysis (PCA) for dimensionality reduction and to investigate about the performance of these methods based on different datasets. Due to the lack of any general evaluation measure for dimensionality reduction methods, we have tried to compare the methods based on the measure of classification accuracy and the execution time of the models. For this purpose, a baseline classification model has been developed for each dataset using logistic regression by utilizing all the original information of the data. The baseline accuracy is then used as an evaluation measure for comparing the methods performance. The methods have been applied on two different types of real-world datasets: Human activity recognition system (HAR) dataset and MNIST dataset. The results show that PCA performs better both in terms of the classification accuracy and execution time on the HAR. Deep auto-encoder improved the classification accuracy of the logistic model for MNIST data, but it was not able to optimize the time execution of the model. The performance of auto-encoders in terms of the classification accuracy was comparable but their training time was longer. In future as more advancements come in the field of deep learning for optimizing the algorithms execution time, it may outperform other dimensionality reduction methods.

Acknowledgements

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You setup in my mind to improve it. Indeed, I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout in every department of my life. I would also like to express special thanks to my supervisor Dr. Firdos Khan for his help throughout my thesis and also for the Statistical inference and applied stochastic models courses which he has taught me. I can safely say that I haven't learned any other Statistics subject in such depth than the ones which he has taught. I would also like to pay special thanks to the chairman of the Statistics Department Dr. Tahir Mehmood for his tremendous support and cooperation. Each time I got stuck in something, he came up with the solution. Without his help I wouldn't have been able to complete my thesis. I appreciate his patience and guidance throughout the whole thesis.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study. I would also like to thank technical and non-technical staff in the department for their cooperation.

Dedication

Dedicated

to

My exceptional parents and adored siblings whose tremendous support and cooperation led me to this wonderful accomplishment.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

DR	Dimensionality reduction
DRT	Dimensionality reduction techniques
PCA	Principal Component analysis
PCs	Principal Components
LR	Logistic Regression
HAR	Human Activity Recognition system
MNIST	Modified Natioal Institute of Standards and Technology

List of Symbols

π	pi
β	beta
ϕ	Phi
σ	sigma

Chapter 1

Introduction

The amount of data generated by various type of organizations is enormous and very high-dimensional (Gao et al., 2015; Amaratunga and Cabrera, 2016). Visualization and analysis of such high-dimensional data not only needs a lot of time, but the complexities associated with its analysis is not easy to handle (Naik et al., 2008; Verleysen and François, 2005). Therefore, such type of data needs to be compressed and reduced in dimensions to easily analyze the data and optimize the time needed for extracting meaningful information from it (Mateen et al., 2009).

Dimensionality reduction techniques (DRTs) provide an effective way to compress the number of input variables before applying any machine learning model (Vinay et al., 2005). DRTs are also used as a data pre-processing step in different machine learning (ML) model to optimize the performance of ML model (Obaid et al., 2019; Sorzano et al., 2014). The process of dimensionality reduction reduces the data dimensions through feature extraction and feature selection. Feature extraction reduces the data dimensions by projecting data into a new low dimensional space forming a new set of variables or dimensions which reduces the noise and redundancy in data while feature selection reduces the data dimensions by selecting only a subset of the original data dimensions which are most relevant to the data. Each DRT reduces the data dimensions either through feature extraction or feature selection.

Different types of dimensionality reduction techniques exist in literature for the purpose of eliminating irrelevant and redundant features (Mateen et al., 2009; Cao et al., 2003). Each DRT reduces data dimensions in a different way. DRTs can be broadly classified into two categories: linear and non-linear DRTs. Linear DRTs reduces dimensions of the data by finding a linear transformation of the data. Linear DRTs try to find a linear structure in the data along which most of the data variation is present and project the data onto it while non-linear DRTs transforms the data from higher dimensional space into a lower dimensional space by finding a non-linear transformation. Most of the data contains non-linear structure and linear DRTs are unable to reduce this type of data in a meaningful way therefore different non-linear DRTs have been developed to reduce the data dimensions efficiently (V.S and Surendran,2015). Example of linear DRTs include Principal component analysis (PCA), Independent component analysis (ICA), Linear discriminant analysis and Singular value decomposition (SVD) etc. Non-linear DRTs include kernel PCA, Isomap, multidimensional scaling, T-stochastic neighborhood embedding etc. All these techniques used for the purpose of data reduction have their own merits and demerits. Some of them may perform better in reducing the data for visualization purposes and some of them may perform better in reducing the noise and redundancy in the data. Therefore, before doing data reduction of any specific dataset and for a given specified task we must compare different DRTs to investigate which techniques is the most optimal for the required task.

Selection of the best suitable DRT can enhance the efficiency of the algorithm and decrease the response time of the model but the major difficulty with applying DRTs is the selection of any specific technique because every technique is developed to covers some specific aspects of the data (Ayesha et al., 2020). A particular dimensionality reduction (DR) technique may be significant for one type of data but not suitable for another type of data. DRTs lacks an agreed upon evaluation measure upon which we can determine which method is best suitable for dimensionality reduction (Lewis et al., 2012; Vantuch et al., 2016). Therefore, in literature mostly DRTs are compared based on some specific characteristics such as computational complexity, adjustable parameters, and classification accuracy (Anowar et al., 2021; Cox et al., 2005).

Comparison of different dimensionality reduction techniques offers an efficient way to select a best suitable method for a given data set (Vantuch et al., 2016; Fournier and Aloise, 2019). It helps to enhance the processing speed and reduce the time needed for discovering meaningful information from the data.

PCA is the most widely used traditional method for dimensionality reduction (Hotelling ,1933). The basic idea of data reduction under PCA is to transform the high-dimensional data space into a low-dimensional data space by forming a new set of dimensions as a linear combination of the original variables (Salem and Hussein, 2019; Cao et al., 2003). These new set of dimensions are uncorrelated with each other and called as principal components (PCs). The main purpose is to accumulate all the original variables which are correlated with each other into their respective principal components along the maximum variance direction so that most of the data variation is contained in the first few principal components (Hasan and Abdulazeez, 2021; Jolliffe and Cadima, 2016). PCA informs us about the variation explained by each individual principal component and hence we can get an idea that how many principal components is needed for explaining 90-95 percent of the data variation. Because PCA is considered as the most fundamental method of dimensionality reduction, therefore, in this research paper we have tried to compare PCA with some of the DR techniques of deep learning such as auto-encoders to discover their dimensionality reduction abilities.

Auto-encoders are type of neural networks consisting of three main layers: an input layer, a hidden layer, and an output layer (Wang et al., 2015). The basic idea of auto-encoders is to compress the input data into some meaningful representation and then reconstruct the original input variables from this representation (Wang et al., 2015; Wang et al., 2014). The compression of data is done by the hidden layer also called as the bottleneck layer (Hinton and Salakhutdinov, 2006). The architecture of auto-encoders is very important because based on the type of architecture the auto-encoders can be classified as simple auto-encoders with only one hidden layer and deep auto-encoders with many hidden layers stacked upon one another (Wang et al.,2015; Zhang et al., 2018). Using auto-encoders for dimensionality reduction, the number of hidden layers nodes is very important because it is the stage where data reduction happens (Wang et al., 2015; Sakurada and Yairi, 2014). Keeping the number of hidden layer nodes less than the original input variables of the data, we can get a compressed representation of the data. PCA and auto-encoders reduces the data dimensions nearly the same way, but PCA differs from auto-encoders in two ways: the first one is that PCA uses a linear transformation while auto-encoders uses a non-linear transformation, the second one is about the ordering of the axes. In PCA, the axes are ordered with respect to the variation explained by them, while in auto-encoders there is no such ordering of axes (Ladjal and Newson, 2019).

This paper attempts to compare PCA with auto-encoders based on the classification accuracy of the classification model and the time of execution. We have used logistic regression as our baseline classification model. A baseline classification accuracy is recorded for the datasets by using all the original dimension as the model input. PCA and auto-encoders are then applied to the given datasets and the model is being trained on the reduced datasets. Evaluation of the applied methods is then performed by comparing the classification accuracy of the proposed methods and time execution with the baseline model.

The main objective of this research study is to compare deep learning methods used for dimensionality reduction such as auto-encoders to the traditional methods such as PCA on some specific types of data sets and then investigate that which method is more suitable for the dimensionality reduction of these data sets. The comparison could help us discovering the dimensionality reduction abilities of auto-encoders and save the time needed for selecting a best suitable method for dimensionality reduction. The experiments to compare the mentioned methods were conducted on two types of datasets: one is about the Human activity recognition (HAR) system and the other one is about the Modified National Institute of Standards and Technology (MNIST) digit recognition system.

The rest of the thesis is organized as: chapter 2 presents the literature review, chapter 3 presents the dataset description; chapter 4 discusses methodology and methods; chapter 5 presents data analysis and results while chapter 6 presents discussion and conclusions.

Chapter 2

Literature Review

Dimensionality reduction is an important topic in the field of data analysis. Different types of techniques and methods have been developed for reducing data. All these techniques contribute to the development of data analysis and making data analysis faster and more productive. Researchers are trying to optimize the existing DRTs and develop new methods to help select the best method in shorter time. Different researchers have tried to compare DRTs based on the structure and characteristics of a datasets to investigate which method is best for DR. Here, we review some of the previous work related to DR.

(Mateen et al., 2009) in their research study have compared twelve most important DRTs with the famous traditional method PCA to investigate to what extent these non-linear DRTs can outperform PCA and to identify their weaknesses. They have compared PCA with (1) Kernel PCA, (2) Isomap, (3) Maximum Variance Unfolding, (4) diffusion maps, (5) Locally Linear Embedding, (6) Laplacian Eigenmaps, (7) Hessian LLE, (8) Local

Tangent Space Analysis, (9) Sammon mapping, (10) multilayer autoencoders, (11) Locally Linear Coordination, and (12) manifold charting by evaluating their performances both empirically and theoretically.

The comparison has been performed by applying all these methods on both artificial and real-world datasets. Three types of characteristics have calculated for all the methods on both artificial and real-world dataset and compared which include generalization error of the classification model, trustworthiness, and continuity of the low dimensional embeddings. Trustworthiness refers to the proportion of points that are too close to one another in the low dimensional space. The classifier used for measuring the generalization error was 1-nearest neighbor classifier. All the above three measurement evaluate the local structure of the data. These measurements investigate to what extent the local structure of the data has been retained.

The experiments have been performed on five artificial datasets which are: (i) The Swiss roll dataset, (ii) Helix dataset, (iii) the Twin peak dataset, (iv) the broken Swiss roll dataset (v) High-dimensional dataset (HDD) and the five real world datasets are (i) the MNIST dataset, (ii) the COIL20 dataset, (iii) the NiSIS dataset, (iv) the ORL dataset, (v) the HIVA dataset. The MNIST dataset is dataset of handwritten images, and the dimensions of the data is equal to 784. The COIL20 dataset consists of images of 20 different objects with dimensions equal to 1,024. The NiSIS dataset is a dataset of pedestrian detection images. There is total 3,675 greyscale images of size 36*18 pixels with dimensionality equal to 648. The ORL dataset is a face recognition dataset containing 400 images with size 112*92 pixels and the dimensionality equal to 10,304. The HIVA dataset is dataset about the drugs discovery consisting of 3,845 points with dimensionality equal to 1,617.

The results of the experiments on artificial dataset shows that the techniques which are based on the neighborhood graphs such as Isomap, MVU, LLE, Laplacian Eigenmaps, Hessian LLE and LTSA, performs strongly on the Swiss roll dataset while the techniques which do not utilize the neighborhood graphs such as PCA, diffusion maps, kernel PCA, Sammon mappings and auto-encoders performs poorly on the Swiss roll dataset. Similarly evaluating the performance on other artificial datasets based on the three measurements, the author has concluded that overall, the techniques that are manifold learner perform well on the artificial dataset. The results of the experiments on the natural dataset shows that manifold learners were unable to achieve the same results. The authors have shown that on natural dataset, the techniques which are not manifold learner such as PCA, Sammon mapping and auto-encoders perform good.

Their conclusion shows that non-linear techniques were unable to outperform PCA. They have showed that the main weakness of non-linear DRTs is that they only perform on selected types of datasets.

(Ayesha et al., 2020) in their research study have discussed both linear and non-linear DRTs and presented a comparative study. The study presents different types of linear and Non-linear DRTs along with their variants. The linear DRTs include PCA, Singular value decomposition (SVD), Latent semantic analysis, Locality preserving embeddings, independent component analysis (ICA) and projection pursuit. The non-linear DRTs discussed include Kernel-PCA, multi-dimensional scaling, Isomap, locally linear embeddings, learning vector quantization, self-organizing map and T-stochastic neighborhood embeddings (T-SNE).

The authors have analyzed these different DRTs by applying it on a dataset which consist of ElectroCardioGram (ECG) signals for heartbeat data with number of observations and dimensions equal to 1094446 and 188 respectively. They have applied the techniques on the datasets to reduce the data dimensions and visualizes only the first four components using scatter plot. They have compared the techniques based on execution time and showed that non-linear DRTs such as Isomap, MDS and t-SNE have higher time of execution.

The authors have discussed some of the issues and challenges while applying DRTs. They have discussed that selection of an appropriate DRT is time consuming and difficult. The other main issue with DRT is the noise present in the data. In most of the cases it is difficult to identify the noise present in the data. Redundancy in the data is also an important issue with DRT. It is difficult to decide how much redundancy should be removed from the data without affecting the performance of the method. Authors have shown that one of the common problems with DRT is to decide how much dimensions should be reduced and how many features to select for the data analysis.

The authors have concluded that the mostly DR to achieve good results need human collaboration. They have also showed that linear techniques take lesser time of computation while non-linear techniques are time consuming, but they are helpful in reducing complex datasets.

V.S and Surendran, (2015) have also discussed different linear and non-linear DRTs. The paper has discussed PCA, Independent Component Analysis (ICA), Canonical Correlation Analysis (CCA), Singular Value Decomposition (SVD), CUR Matrix Decomposition, Compact Matrix Decomposition (CMD), Non-negative Matrix Factorization (NMF), Linear Discriminant Analysis (LDA), Kernel PCA, Multi-dimensional Scaling (MDS), Isomap, Locally Linear Embeddings, Laplacian Eigen map, Local Tangent Space Alignment and Fast map. The authors have compared all these techniques and concluded that each technique has some limitations and cannot be used for every type of data reduction. The authors have concluded that for PCA to provide good and effective results, the data dimensions should be correlated with each other. PCA perform poorly on uncorrelated data while for ICA to be effective, the data should be uncorrelated because ICA performs better on uncorrelated data. The limitation of ICA is that the components generated may not be relevant to the data. Authors have discussed that SVD produces a good approximation of the data with minimum reconstruction error, but the vectors formed may be unable to define its meaning in terms of the data from which it has been drawn. CUR and CMD are good at selecting columns and rows from the data to eliminate the interpretability problem. LDA reduces data dimensions by finding directions along which classes are most separable. Kernel PCA reduces data dimensions by considering the kernel function. Its performance depends on the selection of an appropriate kernel.

MDS can perform both linear and non-linear DR by using its different variants. MDS is helpful for data visualization. LLE, Isomap and Fastmap are considered as variants of MDS.

Vantuch et al, (2016) have compared different DRTs based on statistical dependencies. The authors have compared four different DR methods which include PCA, Non-negative Matrix Factorization (NMF), Auto-encoders and Neighborhood Preserving Embedding. The methods were compared based on the time series of reconstructed dataset (dataset after reduction) and the time series of the original dataset. The experiment has been performed on a dataset which consists of stock market prices. The data is a time series data and contain 1548 rows and 1500 columns. The rows represent time series of 387 investment symbols such as low, high, open, and close prices. The columns represent daily observations. Authors have compared the time series of the reconstructed dataset and the original dataset based on different statistical dependencies. The simple comparison was performed using Euclidean distance, which measures the distance between two points in an N-dimensional space. The three other statistical dependency measures which the authors have used include the correlation coefficient, mutual information, and granger causality.
Correlation coefficient calculates the covariance between X and Y divided by the standard deviation of their product. Mutual information is a type of non-linear evaluation which shows the effect of one random variable over another in terms of the level of uncertainty while granger causality evaluates the causal interaction between two time series.
The authors have concluded that the method of manifold learning and auto-encoders were unable to maintain statistical dependencies of the reconstructed dataset while PCA and NMF both performs better in keeping the statistical dependencies.

Dimensionality reduction also helps in enhancing machine learning models. Reddy et al., (2020) have investigated the performance of two famous methods PCA and LDA on machine learning models. The authors have applied both methods on three different datasets and compared their performance. PCA performance was better than LDA on the Cardiotocography and IDS dataset while on Diabetic dataset the performance of both PCA and LDA was negative. The authors have suggested that in future these methods may perform better on high datasets such as images and text datasets.

Obaid et al., (2019) in their research study have applied different DR techniques to check the impact of data pre-processing and DR on the accuracy of machine learning models. They have used PCA and LDA as data reduction methods and evaluate the performance by checking the classification accuracy of the model. The authors have shown that both PCA and LDA were helpful in reducing the data and getting high accuracy in minimum time.

One of the main difficulties with DRTs is how to choose which DRT is best and optimal for a given problem. The authors have tried to investigate whether humans are consistent with their evaluation of embeddings when the embeddings are different and what structures human are looking for in the dataset to evaluate the embeddings. They have concluded that for a DRT to be selected wisely, it is important to look for a specific data structure that can enhance their speed of algorithms.

Due the evaluative vacuum present in DRTs, mostly DRTs are compared based on the different characteristics to look for an optimal and best DRT. Fournier and Aloise, (2021) have performed an empirical comparison between auto-encoders and traditional DRTs to select a best suitable method. The authors have compared PCA which is the most famous traditional method of DR with two different variants of auto-encoders which is deep auto-encoders and variational auto-encoders and Isomap. The authors have performed the experiments on three different image datasets: MNIST, Fashion-MNIST and CIFAR-10. They have concluded that the accuracy of auto-encoders and PCA are comparable, however, auto-encoders takes higher computation time.

Almotiri et al., (2017) have compared PCA and Auto-encoders for the purpose of E-learning using handwritten digits. Their results showed that auto-encoders got a little bit higher accuracy for the model than PCA, however, PCA was faster than auto-encoders in terms of the execution time.

Wang et al, (2015) have used auto-encoders for dimensionality reduction to investigate its DR abilities. The authors have applied auto-encoders on different datasets and compared with other DRTs. They have concluded that auto-encoders are good choice for DR and for repetitive data structures auto-encoders are more suitable than other DRTs because it can detect repetitive structures.

Chapter 3

Dataset Description

We have used two types of real-world datasets in our analysis. One is the Human activity recognition system (HAR) dataset while the other one is MNIST (Modified National Institute of Standards and Technology) digit recognition system dataset consists of handwritten images. Both the datasets were downloaded from Kaggle(www.kaggle.com). The detail about each dataset is given in the subsequent subsections.

3.1 Human activity recognition system

Human activity recognition is the process of identifying a person's activity based on the senor readings. HAR can be helpful in identifying an abnormal activity, a threating situation and a person in need of assistance. HAR is a dataset made from the recordings of human activities performed by 30 participants carrying out waisted-mounted smartphones. The recordings were taken from the sensors attached to the phone. The objective of the dataset is to classify the data based on the six different activities performed which include (walking, walking upstairs, walking downstairs, sitting, standing, laying). The dataset contains 563 columns in total in which one of the columns is the subject ID performing the task while one column is about the label of the activity performed. So, dropping these two columns we are left with 561 columns which are considered as dimensions of the data set. The dataset contains 7352 rows in the training data set while 2947 in test data set.

The HAR dataset is a time series dataset in which the readings were recorded from the embedded sensors in the Samsung galaxy S II which every participant was wearing on the waist. The accelerometer and gyroscope were used to capture the three axial linear acceleration and three-axial angular velocity at a constant rate of 50Hz. The dataset was then randomly partitioned into test and train datasets, where 70 percent of the data were used for training and 30 percent for test.

The raw data was cleaned, and human feature engineering approaches were applied to extract the time and frequency related features from the data that has been used in the field of human activity recognition. The pre-processing of data was performed in two steps:

1. Noise filters were applied to the gyroscope and accelerometer data 2. Splitting the data into fixed sized window of 2.56 seconds (128 data points) with50In this way a total of 561 features were obtained. The features include both body and motion related components.

3.2 MNIST digit dataset

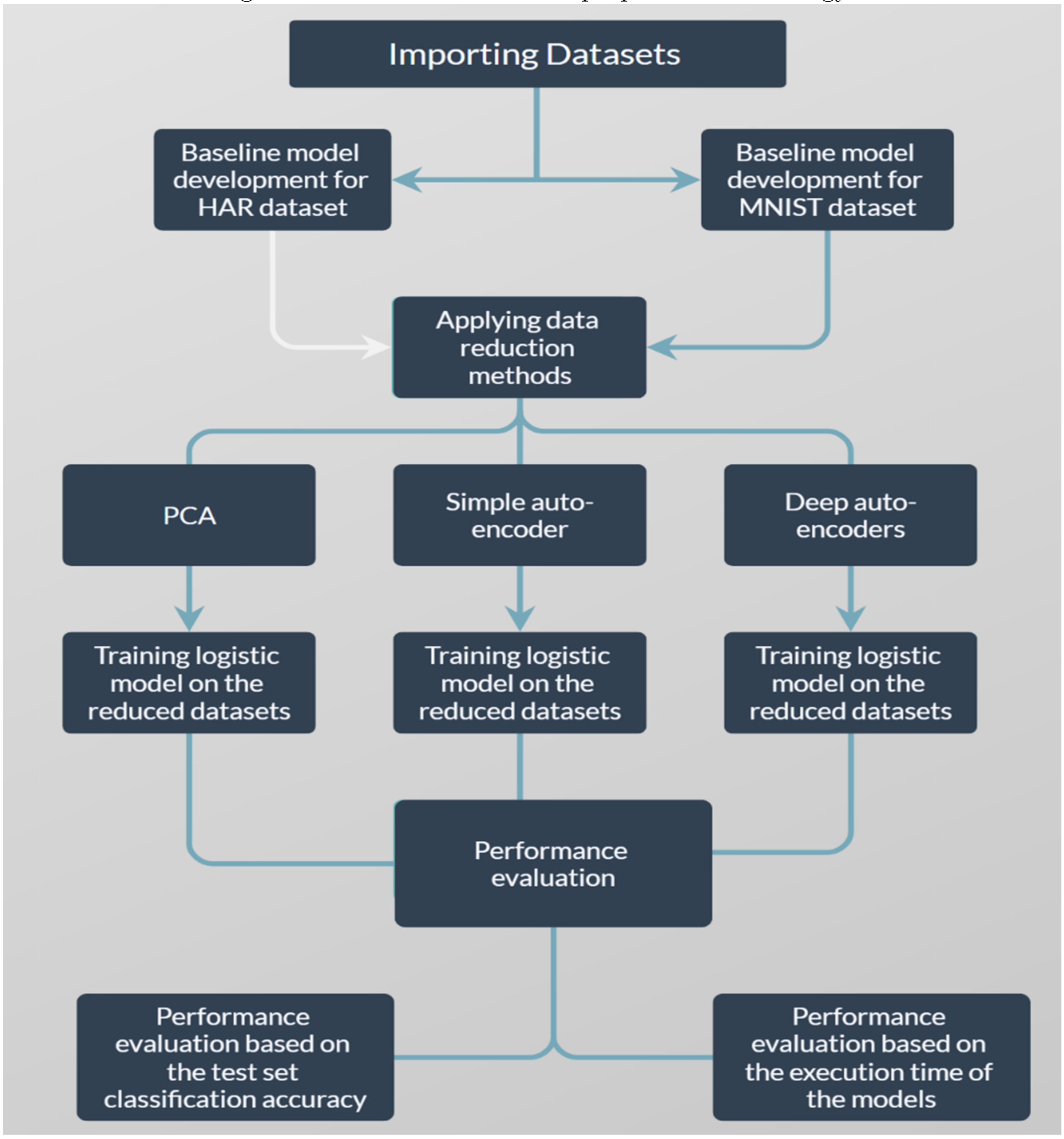
MNIST digit dataset is a dataset of handwritten digits images. The goal of this dataset is to correctly identify digits from a set of tens of thousands of handwritten digits images . The dataset consists of handwritten images from 0 to 9 each of pixel 28*28 for a total of 784 pixels. The pixel is an integer between 0 and 255. The training data set consists of total 785 columns in which one of the columns represent the 'label' of the digit while all other 784 columns represent the pixel values which are considered as dimensions of the dataset. The training set contain 42000 rows while the test set contain 28000 rows. The test dataset here does not contain the 'label' column and we cannot measure the test set classification accuracy therefore we have split the training data into the ratio of 60:40 with 60 percent for training data and 40 percent for test data.

Chapter 4

Methodology

As we have mentioned in the introduction section that dimensionality reduction lacks an agreed upon evaluation measure upon which we can decide which method is better. Every method used for dimensionality reduction has its own merits and demerits. A method may be suitable for visualization purpose but not for classification. Therefore, for evaluating the performance of different dimensionality reduction techniques, some specific characteristics related to dimensionality reduction methods are used as evaluation measure. We have used classification accuracy and execution time of the models as evaluation measure for comparing the methods. The methods have been applied on both the datasets and the results are compared to the baseline model and with each other. For the evaluation of the performance of the proposed methods, a baseline classification model is first developed for both the datasets by utilizing all the original information. Logistic regression has been used for developing the baseline classification model. The flowchart for our proposed methodology is given in Figure 4.1.

Figure 4.1: Flow chart for our proposed methodology


Importing Datasets
Baseline model development for HAR dataset
Baseline model development for MNIST dataset
Applying data reduction methods
PCA
Simple auto-encoder
Deep auto-encoders
Training logistic model on the reduced datasets
Training logistic model on the reduced datasets
Training logistic model on the reduced datasets
Performance evaluation
Performance evaluation based on the test set classification accuracy
Performance evaluation based on the execution time of the models

4.1 Logistic regression

Logistic regression is one of the most important statistical techniques used in data analysis for the classification of categorical variables. The advantage of using logistic regression is that it is simple and provide probabilities that can also be extended to a multi-class classification problem (Maalouf, 2015). The basic mathematical concept that logistic regression utilizes is the logit transformation of the dependent variable. The advantage of using logit function in logistic regression is that it maps probabilities to the set of real numbers. Logit is a link function that can be used to transform the probabilistic values of a categorical variable into continuous values. It is equal to natural log of odds of the response variables. Let we have a response variable Y with two classes either 0 or 1 the

$$logit(Y) = naturallogofodds = \ln\left(\tfrac{\pi}{1-\pi}\right) = b_0 + b_1x_1 + ... + b_rx_r$$

Odds represent the ratio of the probability of success to the probability of failure for a given variable.

$f(x) = b_0 + b_1x_1 + ... + b_rx_r$

represent the linear logistic classifier also called the logit function. So, if for a given class x, $p(x)$ represent the probability of the class then

$$f(x) = \ln\left(\frac{p(x)}{1-p(x)}\right) \tag{4.1}$$

here we have used $p(x) = \pi$ for the two class variable.

$\pi = Probability(Y =$outcome of interest$|X = x,$a specific value of x$) = \left(\frac{e^{(b_0+b_1x_1+....+b_rx_r)}}{1-e^{(b_0+b_1x_1+....+b_rx_r)}}\right)$

$$\pi = \left(\frac{e^{(b_0+b_1x_1+....+b_rx_r)}}{1-e^{(b_0+b_1x_1+....+b_rx_r)}}\right) \tag{4.2}$$

This is the logistic regression equation for a binary outcome variable. Here $b_0, b_1, ...b_r$ represent estimator of the regression coefficients. The coefficients are estimated through the method of maximum likelihood estimation which means to get the best possible estimate we maximize the log-likelihood function for all the observations $i = 1, 2, ...n$.

Here one thing should be noted that logit is a link function which transform a given parameter and it is equal to logarithm of the odds while logistic function is the inverse of the logit. So, in the above formula $f(x)$ represent the logit function while $p(X)$ is the logistic function given by

$$p(x) = \left(\frac{1}{(1 + exp(-f(x))}\right) \tag{4.3}$$

For a response variable with multi-classes, we use multi-class logistic regression. To apply multi-class logistic regression the following steps are performed Let X represent the data matrix, Y represent the vectors of classes and W be the weight matrix, then first we calculate the product of X and W. Let

$Z = XW$

Then we take the SoftMax of each row of Z. SoftMax is the transformation function used for transforming categorical variables just like we used logit for two class variables.

$$P_i = softmax(Z_i) = \frac{exp(Z_i)}{\exp\left(\sum_{k=0}^{k} Z_{ik}\right)} \tag{4.4}$$

Each row of Z should be the product of each row of X and the whole matrix W. Also, each row of P should add to one. Thirdly, we take the argmax of each row and find the class with highest probability. In this case we estimate set of K weight vectors w1,.......wk and biases b.

The two most common way to estimate the parameters are

1. Least Squares Optimization (LSO)
2. Maximum Likelihood Estimation (MLE)

But for logistic regression MLE is used to estimate the parameter because the cost function with SLO is difficult to minimize and finds the solution for logistic regression. The complexity and non-linearity of the cost function makes it difficult to find the optimal solution. MLE is a parameter estimation probabilistic framework that tries to estimate the best possible parameter by finding the maximum likelihood function. The basic idea is to maximize the conditional probability of observing Y given X under the logistic model. As logistic regression predicts probabilities of the given classes so we can find their likelihood function. For each training example of the data, that is a vector of independent variables given by, xi, and the corresponding class, yi, the likelihood function is given by As the probability of the class was either "p", if y=1, and 1-p, if y=0 ,the likelihood is therefore

$$L(\beta_0, \beta) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i)^{1-y_i} \tag{4.5}$$

The log-likelihood is equal to

$$\ell(\beta_0, \beta) = \sum_{i=1}^{n} y_i \log p(x_i) + (1 - y_i) \log 1 - p(x_i) \tag{4.6}$$

$$= \sum_{i=1}^{n} \log 1 - p(x_i) + \sum_{i=1}^{n} y_i \log \frac{p(x_i}{1 - p(x_i)} \tag{4.7}$$

$$= \sum_{i=1}^{n} \log 1 - p(x_i) + \sum_{i=1}^{n} y_i(\beta_0 + x_i\beta) \tag{4.8}$$

$$= \sum_{i=1}^{n} - \log 1 - x^{(\beta_0 + x_i\beta)} + \sum_{i=1}^{n} y_i(\beta_0 + x_i\beta) \tag{4.9}$$

Now to find the ML estimates, we take the derivative of the above equation with respect to the parameters and set the derivative equal to zero and solve the problem.

For multi-class logistic regression, we would follow the same procedure but in case of multi-class logistic regression we would find set of k parameters for every class individually. For each class z in the set 0:(k-1) we would find its own $\beta_0^{(z)}$ and $\beta^{(z)}$ with the probabilities given by

$$Pr(Y = z \mid \vec{X} = x) = \frac{e^{(\beta_0^{(z)} + x\beta^{(z)})}}{\sum_c e^{(\beta_0^{(z)} + x\beta^{(z)})}} \tag{4.10}$$

To find the best optimal weights and biases for a logistic regression a cost function is also necessary. For example, if “y” is our dependent variable that we want to predict and “x” our independent variable then we want

$^i = \sigma(W^t + b)$,

where$\sigma(z^i) = \frac{1}{1+\exp^{-z^i}}$

(x^1, y^1).....(x^m, y^m) we want $\hat{y}^i \simeq y^i$

The loss function finds the difference between the predicted and the original value.For example for a single training example the loss function

$$L(\hat{y}^i, y^i) = \frac{1}{2}(\hat{y}^i - y^i)^2 \tag{4.11}$$

$$L(\hat{y}^i, y^i) = -(y^i \log(\hat{y}^i) + (1 - y^i)log(1 - \hat{y}^i) \tag{4.12}$$

1. if $y^i = 1 : L(\hat{y}^i, y^i) = -log(\hat{y}^i)$ where $log(\hat{y}^i)$ and y^i should be close to 1.

2. if $y^i = 1 : L(\hat{y}^i, y^i) = -log(1 - \hat{y}^i)$ where $log(1 - \hat{y}^i)$ and y^i should be close to 0.

the cost function can be find by taking the average of all the m training examples. The best optimal weights and biases are those which minimizes this cost function.

$$J(W, b) = \frac{1}{m}\sum_{i}^{m} L(\hat{y}^i, y^i) = -\frac{1}{m}\sum_{i}^{m}[(y^i \log(\hat{y}^i) + (1 - y^i)log(1 - \hat{y}^i)] \tag{4.13}$$

Logistic regression in python can performed through the following steps: 1. Importing the necessary libraries and packages for logistic regression such as Numpy, Scikit-learn and matplotlib

2. Numpy is used for mathematical computation, Scikit-learn for performing logistic regression and matplotlib for visualization purposes

3. Creating a classification model by importing logistic regression from scikit-learn library

4. Fitting the model on the given dataset

5. Predicting the test with the model

6. Evaluating the model performance by using different measures such as confusion matrix.

We have used multi-class logistic regression as the classification model for both datasets because both datasets consist of more than two classes.

4.2 Principal Component Ananlysis

Principal component analysis (PCA) is a linear dimensionality reduction method works on the principal of linear algebra. The idea was first discussed in the research paper "Analysis of a complex of statistical variables into principal components" written by Hotelling in 1933.

PCA is the most famous and traditional method used primarily for the purpose of dimensionality reduction. The goal of PCA is to find planes and hyperplanes that closely approximates the original data with very less information to be lost. PCA reduces the data dimensions by finding a linear combination of all the correlated variables in the original data set and then projecting it into a lower dimensional data space. The new variables are independent and uncorrelated with each other ordered in the direction of maximum variance.

Dimensionality reduction through PCA is performed through the following steps:

1. Standardization of the original dimensions or variable
2. Computing Covariance matrix
3. Finding Eigenvectors and Eigenvalues
4. Calculating principal components by multiplying eigenvectors with the standardized columns

Standardization of the original variables is necessary because the principal components largely depend on the variances of the original variables. Therefore, to make the contribution of all the original variables equal we scale or standardize them so that they all equally contribute to the variation of the data. mathematically, to find the standardized variables we use the formula

Z= (value of the original variable – mean of the original data)/(standard devaition of the original data)

For example, if we have a data set represented by the matrix "X" with N number of dimensions, where the number of dimensions is equal to the number of columns in the matrix, and K represents the number of rows which is equal to the number of training examples, then if we want to reduce the data matrix dimensions from “N” to “D” where $D << N$, our first step would be to standardize the data matrix by finding the mean of all the columns.

$$\begin{pmatrix} X_1 & X_2 & X_3 & .... & X_n \end{pmatrix}$$

Here $X_1, X_2.......X_N$ reprsent the columns vectors with number of rows equal to K. After finding the mean of all columns we next find the Covariance matrix of the whole matrix by finding the variances and covariances among the varaibles. The covariance matrix of two variables X and Y can be found using the formula

$$Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{x})(Y_i - \bar{y}) \tag{4.14}$$

PCA mainly finds a linear relation between the original variables and all those variables which are highly correlated to one another are combined to one single principal components. The covariance matrix finds the same relationship between all the variables. The third step involves finding the Eigenvectors and Eigenvalues of the covariance matrix. The basic idea of principal components is that to find new variables that uncorrelated with each other and explains most of the variation of the data. The number of PCs constructed are the same as the number of the original variables but because PCs are arranged in the direction of the maximum variance therefore only those PCs are maintained which explains most variation of the data. Eigenvectors and eigenvalues of the covariance matrix gives us the required principal components because eigenvectors of the covariance matrix actually represents the direction of axes along which maximum variance is contained and the eigenvalues represent the coefficients or the amount of variance contained in each eigenvector. The final step involves recasting the original data along the new dimensions by multiplying eigenvectors with the standardized original data. Mathematically, for a dataset X with N dimensions, the PCs can be calculated using the formula

$$Y = WX$$

Y represent the principal components; W represent the weight matrix which can be calculated from the eigenvectors of the covariance matrix and X represents the standardized columns of the original data.

$$y_{ij} = w_{1i}x_{1j} + w_{2i}x_{2j} + ... + w_{pi}x_{pj}$$

Here i represent the number of principal components and it vary from 1 to p while j represents the number of observations. These are called as principal components and they are ordered in terms of the variance explained by each individual principal component. The component containing most of the data variation is the first principal component, the second principal component explains a data variation less than the first one and greater than the remaining principal components and so on. PCA gives the same number of principal components as the original dimensions of the data but only those principal components are retained which explains maximum variation in the data and the components explaining very less variation can be discarded. Mostly for dimensionality reduction through PCA, only those principal components are retained which can explain about 90 to 95 percent of the data variation. In this research study, 95 percent variance threshold has been set for the number of principal components.

Following are the steps to perform PCA on a dataset in python 1. Importing the required libraries and packages such as sklearn

2. Apply PCA on the training datasets by specifying the number of components you need to be retained

3. Fitting the model on test datasets to reduce dimensions of the test data

4.3 Auto-encoders

Auto-encoders belongs to the class of deep learning algorithms used for dimensionality reduction. Auto-encoders are three-layer neural networks designed to learn and extract meaningful information from the data (Wang et al., 2015).The architecture of auto-encoders was first introduced by (Rumelhart et al., 1986) for the purpose of learning the internal representation to solve the problem of error propagation. They have used hidden layers or units in their neural networks to learn an internal representation of the data which can be meaningful in reducing the error of the model. The idea is very interesting because it can find meaningful structures within the data in an unsupervised manner.

The basic idea of auto-encoders is to take an input x, project it into a meaningful representation and then predict the same x from this meaningful representation. The input layer takes the input data, the hidden layer converts this data into a meaningful representation while the output layer makes the original input data from the compressed representation. The middle-hidden layer is also called as bottleneck layer because it forces the data to a compressed representation. The compressed representation is then used to reconstruct the original input. The difference between the original input data and the reconstructed output data is called the reconstruction error. The model seeks to reduce the reconstruction error to train the model efficiently.

Due to the recent developments in the field of deep learning and the rise of big data, the application of auto-encoders is becoming wider. It has gained an important center place in the "deep architecture" approach, in which auto-encoders are trained for different purposes in the form of Restricted Boltzmann Machine (RBM) to tune the model. These architectures are shown to have achieved state of the art results for different types of problems. Mathematically, a general auto-encoders framework can be represented by

$$\phi : X \hookrightarrow F \tag{4.15}$$

$$\psi : F \hookrightarrow X \tag{4.16}$$

$$\phi, \psi = argmin||X - (\phi \cdot \psi)\,||^2 \tag{4.17}$$

Using this framework different types of architectures and auto-encoders can be formed. Eq. (4.15) and (4.16) represent the most general form of an auto-encoders in which both ϕ and ψ represent a neural network. These neural networks can be of different types and different structures. It depends on the problem on how to make the architecture of the neural networks. In the special case, when both ϕ and ψ are linear functions, auto-encoders becomes a linear auto-encoders, and we get the same latent representation for the original input as PCA. Therefore, auto-encoders are considered as generalization of PCA because PCA can only finds a linear projection space while auto-encoders are able to project data into non-linear space.

Equation (4.15) represents the encoder stage of the auto-encoders in which the auto-encoders takes an input x and maps it into the bottleneck representation $h \in F = R^P$ where

$$h = \sigma(Wx + b) \tag{4.18}$$

the variable h is usually called the decoded variable or latent variable, is the activation function such as we use sigmoid function in logistic regression, W is the weight matrix while b is the bias vector. The weight and biases are usually initialized randomly and during the learning process they are updated iteratively to learn the optimized weights and biases. The equation (4.16) represents the decoder stage in which the latent images or variables are then mapped to the reconstructed input variables $x^{'}$

$$x^{'}=\sigma^{'}(W^{'}\text{h}+b^{'})$$

Here $\sigma^{'}$, $W^{'}$, $b^{'}$are the parameters of decoder stage and may be unrelated to the encoder stage. Equation (4.17) represents the reconstruction loss between the original input and the reconstructed input. Auto-encoders are trained to minimize this reconstruction loss. The reconstruction loss and the activation functions used in the above equations (4.15-4.17) can be different for different types of tasks because auto-encoders are multi-tasks algorithms. It can be used for feature extraction, anomaly detection and information retrieval etc.

The hyperparameters needed to set before training an auto-encoders include the size of the bottleneck layer, the number of hidden layers, the number of nodes per layer and the loss function. Mostly Mean squared error (MSE) loss is used but, in some cases, where the output is a value between 0 and 1 the binary cross-entropy loss is used.

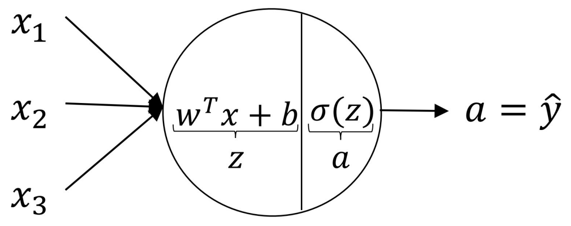
In general, every type of auto-encoders performs DR because auto-encoders finds a compress meaningful representation in the bottleneck layer. This compressed representation is then transformed into the output layer to produce the original input original data. It depends on the architecture of the bottleneck layer on how we want our data to be reduced. Keeping the number of nodes in the hidden layer less than the original input layer we can reduce the data dimensions. The activation functions used in the architecture can be used to find linear or non-linear dimensionality reduction.

We have used two types of auto-encoders in our study, simple auto-encoder (Figure 4.3), and deep auto-encoder (Fig 4.4).

4.3.1 Simple auto-encoder

Simple auto-encoder is a type of auto-encoders in which only one hidden layer is present between the input and output layer. The middle-hidden layer compresses the original dataset and then transfer it to the output layer to reconstruct the original variables. Mathematically, the architecture of simple auto-encoder is the same as that of a simple one hidden layer neural network.

Figure 4.2: A simple neural network with one hidden layer



$$z = w^T x + b$$

$$a = \sigma(z)$$

A simple auto-encoders behaves the same way just like a simple neural network with one hidden layer between the input and output layer. The only difference is that in the auto-encoder architecture the dimensions of the input layer and the output layer is the same while in simple neural network it is not necessary. The input data is given to the network through the input layer. The data is then transferred to the next layer for further processing. The hidden layer multiplies the input variable with a corresponding weights and add a bias factor. The weight vectors and bias factors are randomly initialized. They are updated upon every iteration of the model to learns the best optimal weights. An activation function is then applied to the function to further process it. The activation function is an important step in the neural network because it can be used to prevent linearity. If there were no activation function the results of every layer would be linear function and we would not be able to find any hidden and meaningful information from the data. Not every type of data is linear therefore we need non-linear function to represent these data structure. Non-linear activation functions can be used to project data to non-linear spaces. Some of the commons activation used are sigmoid, Rectified linear unit

(ReLU) and Hyperbolic tangent (Tanh).After the activation function is applied, the data is then transferred to the output layer. The output layer reconstructs the original data from the compressed representation of the hidden layers. The reconstructed data is then compared to the original data by using loss functions. The model seeks to minimize the reconstruction error by updating the weights vectors and bias factors. The weights and biases are updated by the method of backpropagation using gradient decent algorithm. The loss function is calculated using the difference between the original and the reconstructed input. The cost function for m training example is given by

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m}\ell(y,\grave{y}) \tag{4.19}$$

Different types of loss functions are used depending on the type of the datasets. Mean Squared Error loss (MSE), Binary Cross Entropy loss and Categorical Cross Entropy loss are some of the loss functions used in neural networks.

4.3.2 Deep auto-encoder

Deep auto-encoders are another type of auto-encoders in which the number of hidden layers is greater than one. Deep auto-encoders uses many hidden layers which are stacked upon one another to deeply analyze the data. Deep auto-encoders are used for extracting more deep structures in the data. The architecture of deep auto-encoders is same as the deep neural networks with a difference that in case of the deep auto-encoders the number of dimensions of the output layer is the same as the number of input layer.

The mathematics of deep auto-encoders is the same as of simple auto-encoders. Deep auto-encoders only uses many hidden layers to compress the data and finds a meaningful latent representation. The number of nodes of each layer represent the dimensions of that layer. It is an important hyperparameter in the optimization of the network. The python algorithm for applying simple and deep auto-encoder can be performed through the following steps:

1. Importing the necessary libraries and libraries; Tensorflow and keras are python libraries for performing deep learning

2. Making the architecture of the network by defining each layer with the required information.

3. Defining the number of dimensions to which the original dimensions should be reduced.

4. Defining the input layer by giving the number of the original data dimensions as input into the layer.

5. Define the hidden layer by giving the input dimensions and applying the activation function.

6. Define the output layer by giving the number of dimensions from the hidden layer and

applying the activation function.

7. The layers are then combined to form an auto-encoder model which takes the input and maps it into the output.

8. The auto-encoder model is then compiled by applying optimizer and loss function.

9. The model is then fit on the data by specifying the hyperparameters of the model such as the batch size and number of epochs.

All types of neural networks use a method of optimization called the Gradient Decent Algorithm which means the rate of descending. The model is optimized by finding the best possible weights which reduces the value of loss function. For this purpose, the model is iterated again and again to get the optimal results. Gradient Decent (GD) use the derivative approach by finding the derivative of the loss with respect to the hyperparameters by backpropagating through the network. GD uses a hyperparameter called the learning rate which corresponds to the size of the steps taken on the loss curve for finding the derivative. The batch size and number of epochs is also necessary for this iterative process. Batch size represents the number of training examples or samples present in a single batch. Batch size and number of batches are two different things. When the data is too large we can't pass all the data at one once, therefore we divided it into small parts. Theses small parts of the data are called as batches. One epoch is equal to the passing of the whole dataset forward and backward through the network. The entire dataset is passed to the network multiple times to learns more and more and get the optimal results.

Figure 4.3: simple auto-encoder

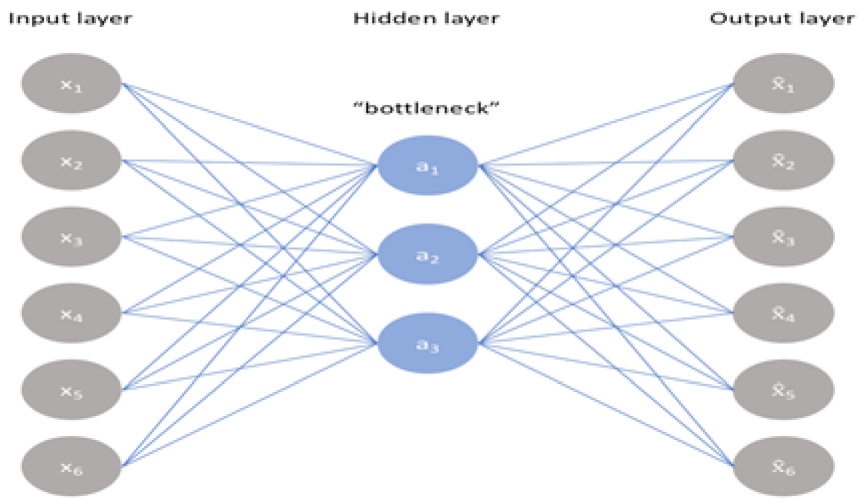
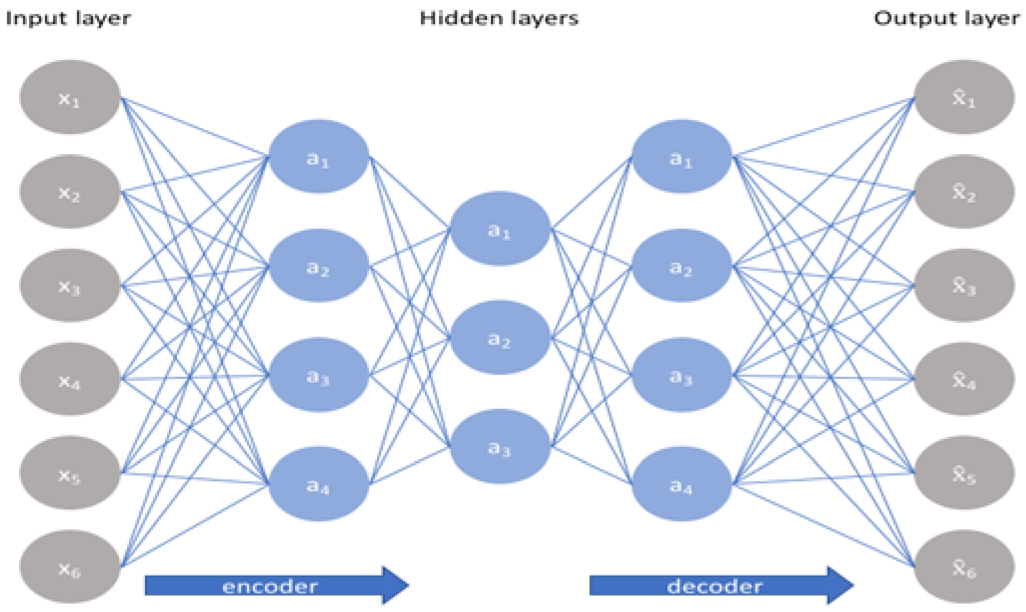


Figure 4.4: deep auto-encoder



4.4 Performance evaluation

The performance of the proposed methods is evaluated based on the test set classification accuracy of the logistic model and the execution time of the methods. To check the performance of the model, we use a simple accuracy score and confusion matrix. Accuracy refers to the fractions of corrected predictions: correct predictions / total number of data points while confusion matrix gives us an insight into the number of correct and incorrect predictions. Confusion matrix is a table showing how many data points are correctly predicted while how many are predicted incorrectly. Time execution of the model refers to the training time taken by methods while learning from the data and compressing it. The execution time of the logistic model after trained on the reduced datasets through the respective methods has also been compared with execution time of the baseline logistic model for both datasets.

Chapter 5

Data analysis and Results

This section explains the results of our experiment and analysis. The data analysis is divided into three parts. In the first part, we develop a baseline logistic model for both datasets using the original data. In the second part, we perform evaluation of our proposed methods based on the classification accuracy of the test data. We apply PCA, simple and deep auto-encoders to compress the datasets and then train the logistic model with the compressed datasets and record the test set classification accuracy. The accuracy is then compared with the baseline model to investigate the performance of the methods. The third part of our analysis compare the applied methods based on the time of execution for both datasets.

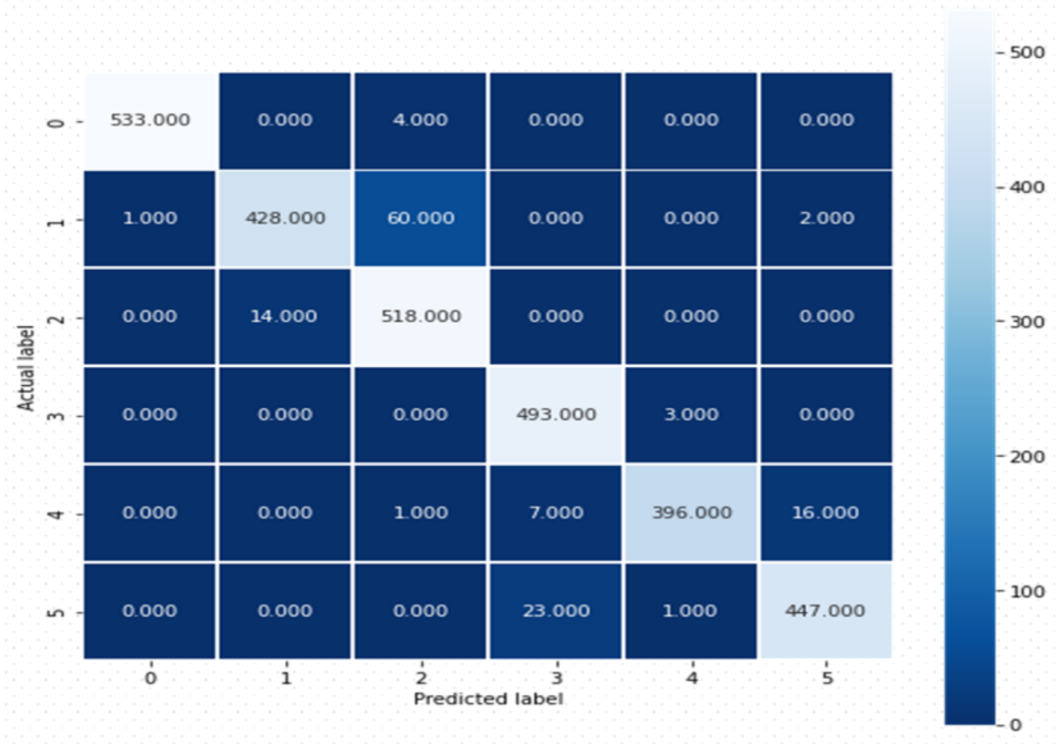
5.1 Development of baseline logistic regression classification model

The baseline model for the datasets is constructed using the multi-class logistic regression model. HAR dataset consists of total 561 columns. These 561 columns are used as input variables in the logistic model to build the baseline model for the performance evaluation. The model was trained on the training dataset and used to predict the test dataset. Evaluating the model classification accuracy on the test data, we get a score of 96 percent. The confusion matrix for the logistic model is given in Figure 5.1.

From the confusion matrix given in Figure 5.1, we can see that about 533 observations were predicted with correct label of 0 and 4 incorrect predictions were made for the label 0. For class label 1, we got 1 incorrect prediction with actual label 1 and predicted 0 while 60 incorrect predictions with actual label 1 and predicted class label 2. The incorrect predictions were a little bit higher for the class label 3 and 1. Overall the confusion matrix shows that the model predictions were better, and we get a good classification accuracy.

The MNIST dataset consists of a total of 784 columns. To develop the baseline model for MNIST dataset we take these 784 columns as our input variables in the logistic model and train the classifier. Training the logistic model by using 784 columns of the MNIST dataset as input variables and classifying the data we get a baseline accuracy equal to 91.61 percent. The confusion matrix is given in Figure 5.2. The confusion matrix shows that the

Figure 5.1: Confusion matrix for the baseline logistic model for HAR dataset.

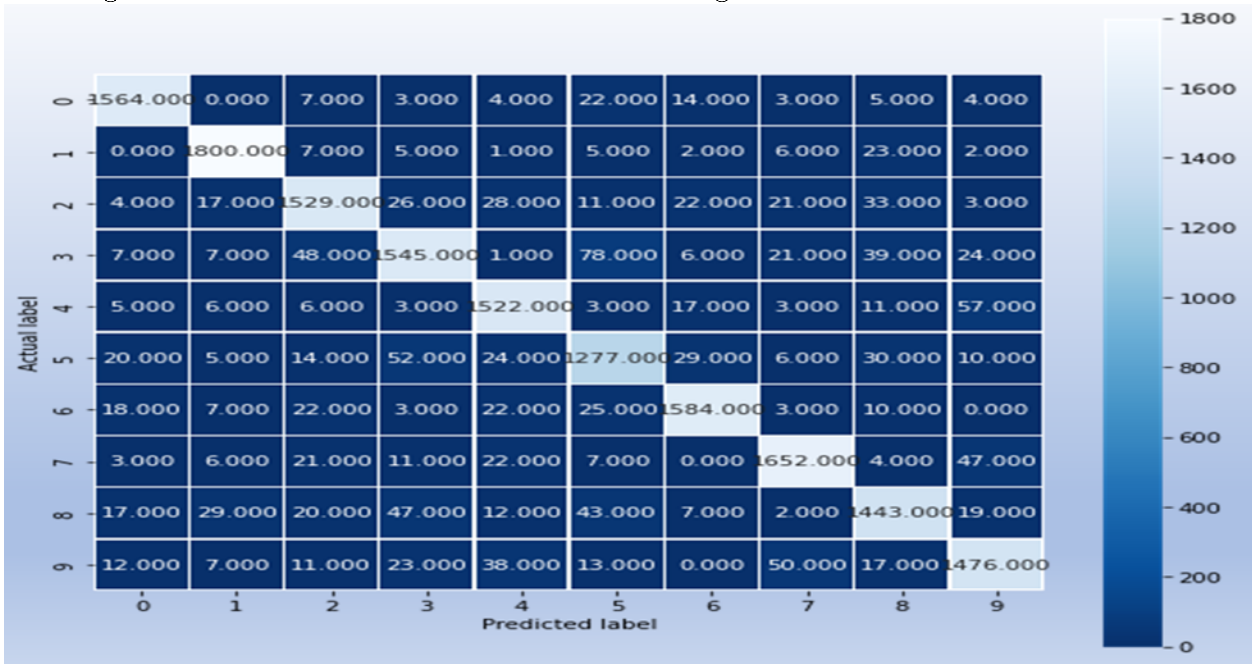


number of incorrect predictions made is larger in number as compared to the HAR model and therefore we get a low accuracy score.

5.2 Evaluation based on the classification accuracy

To evaluate the performance of the methods on the datasets, the methods have been applied on both datasets for reducing the dimensions and then a logistic model is trained on the reduced datasets for each method separately. The classification accuracy achieved from the respective model is then compared with the baseline accuracy based on the test data set.

Figure 5.2: Confusion matrix of the baseline logistic model for MNIST dataset.



	0	1	2	3	4	5	6	7	8	9
0	1564.000	0.000	7.000	3.000	4.000	22.000	14.000	3.000	5.000	4.000
1	0.000	1800.000	7.000	5.000	1.000	5.000	2.000	6.000	23.000	2.000
2	4.000	17.000	1529.000	26.000	28.000	11.000	22.000	21.000	33.000	3.000
3	7.000	7.000	48.000	1545.000	1.000	78.000	6.000	21.000	39.000	24.000
4	5.000	6.000	6.000	3.000	1522.000	3.000	17.000	3.000	11.000	57.000
5	20.000	5.000	14.000	52.000	24.000	1277.000	29.000	6.000	30.000	10.000
6	18.000	7.000	22.000	3.000	22.000	25.000	1584.000	3.000	10.000	0.000
7	3.000	6.000	21.000	11.000	22.000	7.000	0.000	1652.000	4.000	47.000
8	17.000	29.000	20.000	47.000	12.000	43.000	7.000	2.000	1443.000	19.000
9	12.000	7.000	11.000	23.000	38.000	13.000	0.000	50.000	17.000	1476.000

5.2.1 Principal component analysis (PCA)

Before applying PCA, it is important to define the percentage of variance we need for our analysis to be explained by the principal components. To achieve a good classification accuracy, we have used the 95 percent variance threshold for PCA which means that only those principal components will be retained which can explain 95 percent variation in the data. To get an information about how many principal components is needed for HAR dataset for explaining 95 percent variation in the data, we draw a graph showing the relationship between the number of principal components and the cumulative variance.

Figure 5.3 shows that using 95 percent cumulative variance threshold, the curve touches the threshold line approximately at 98 principal components. It means that we need 98 principal components to explain 95 percent of the data variation. We reduce the original data dimensions from 561 to 98 by keeping the first 98 principal components after applying PCA.

The same 95 percent variance explanation threshold has been set for the MNIST dataset. After applying PCA on MNIST dataset, we want to retain only that many number of principal components which can explain 95 percent variation in the data. Figure 5.4 shows the relationship between the cumulative variance explained and the number of principal components needed.
Figure 5.4 shows that approximately 151 principal components are needed to explain 95 percent variation of the original data. So, after applying PCA on MNIST data we retain only 151 principal components which can be used as input dimensions in the logistic model. The logistic model being trained on the reduced HAR dataset gives a test set classification accuracy equal to 93 percent. The confusion matrix for the model plotted in Figure 5.5 represents that the predictions with a reduced number of dimensions is good, however, the classification accuracy is smaller than that of the baseline accuracy.

Figure 5.5 shows that the number of incorrect predictions in this case is a little bit more than the baseline model, therefore, we get a smaller accuracy score for this model than the baseline model. This shows that a smaller number of the original dimensions is correlated with one another and using the 95 percent variance threshold resulted in some loss of information.

Figure 5.3: The relationship between cumulative variance explained and number of principal components for HAR dataset

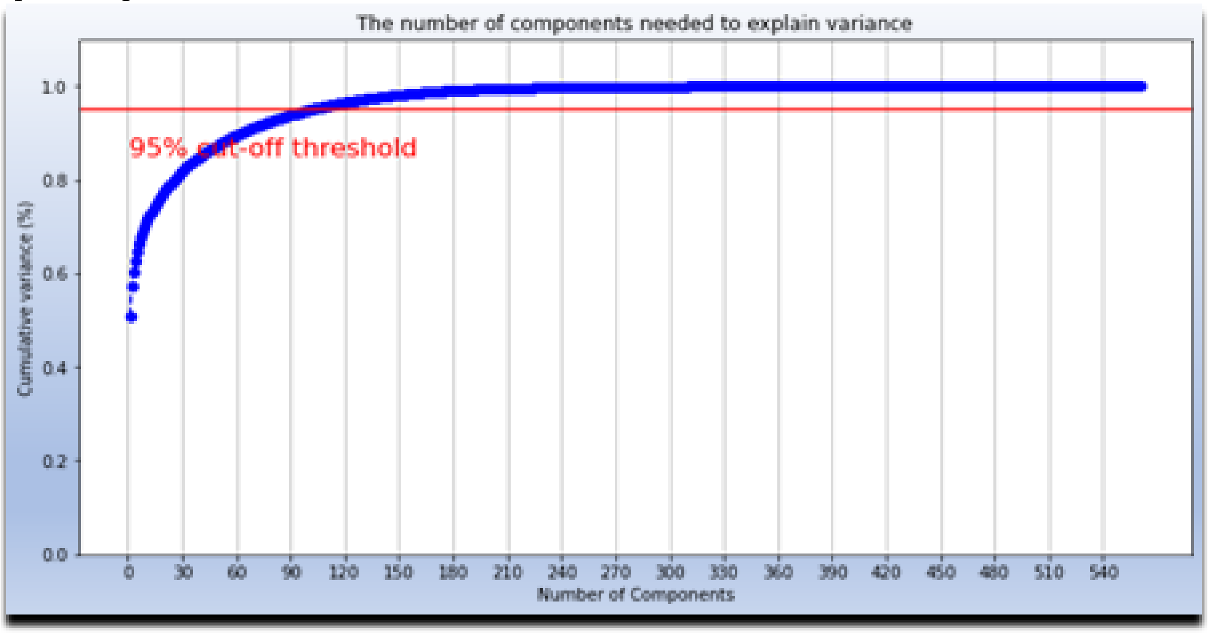


Figure 5.4: A relationship between cumulative variance explained vs the number of principal components for MNIST dataset.

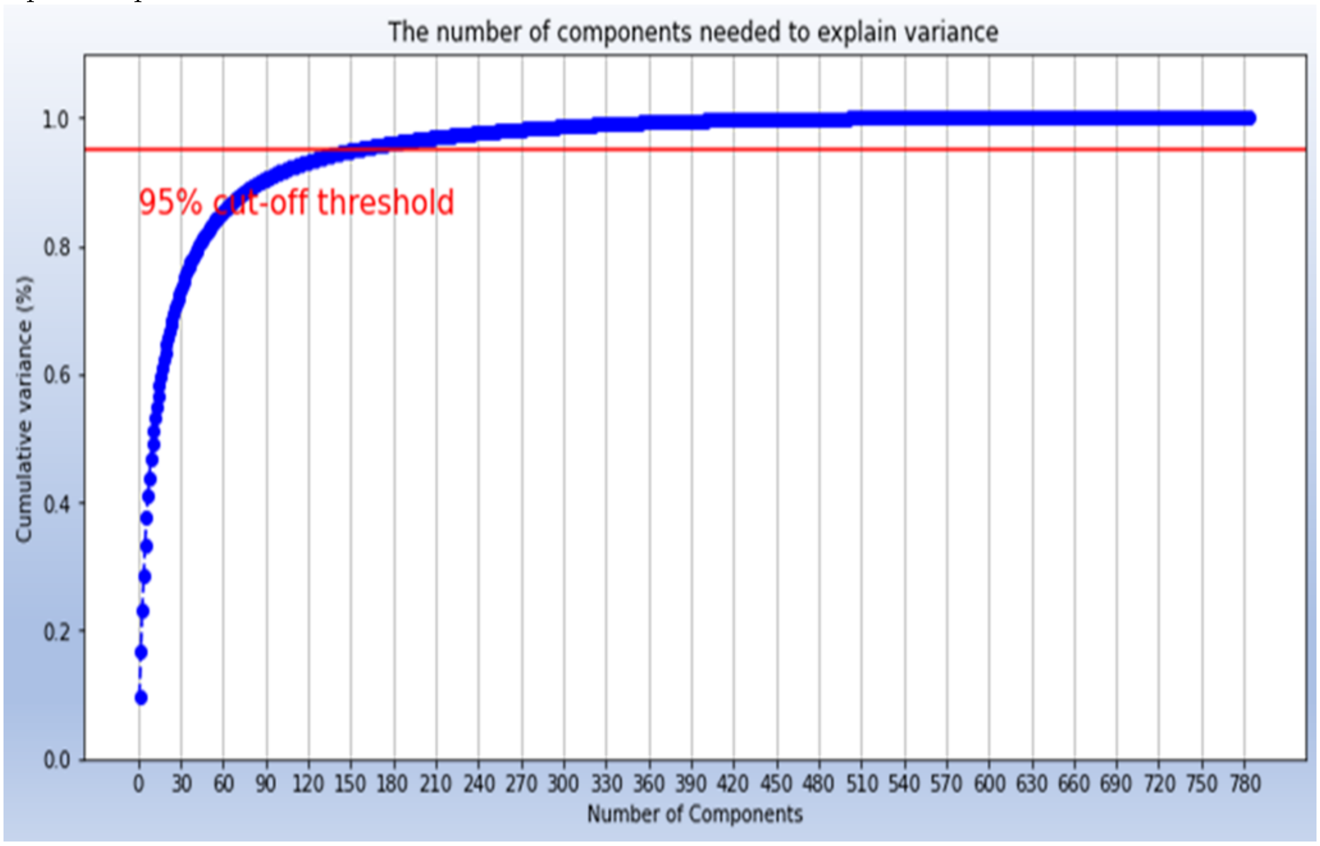
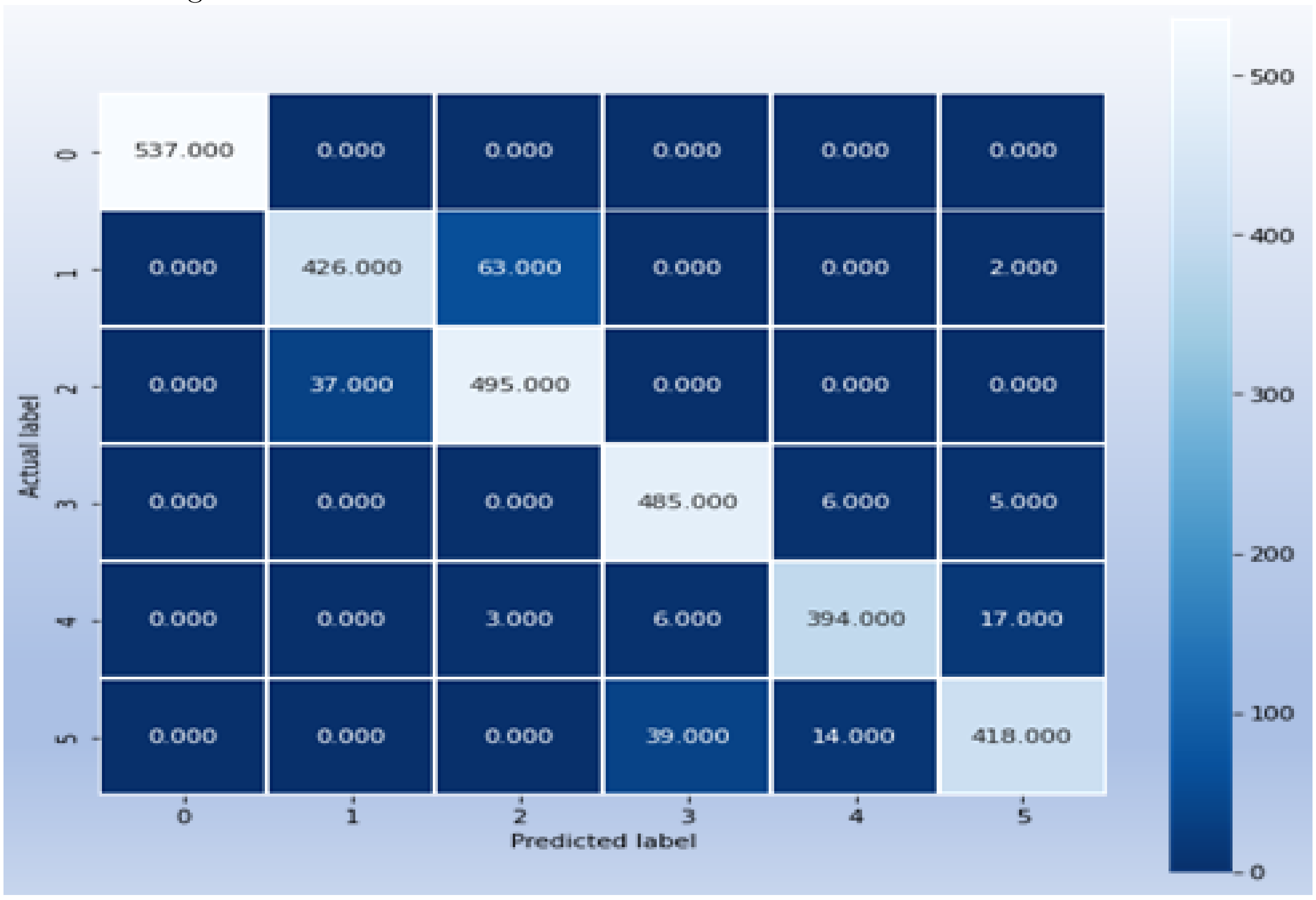


Figure 5.5: Confusion matrix of the logistic model being trained on the reduced HAR dataset through PCA for the evaluation duration

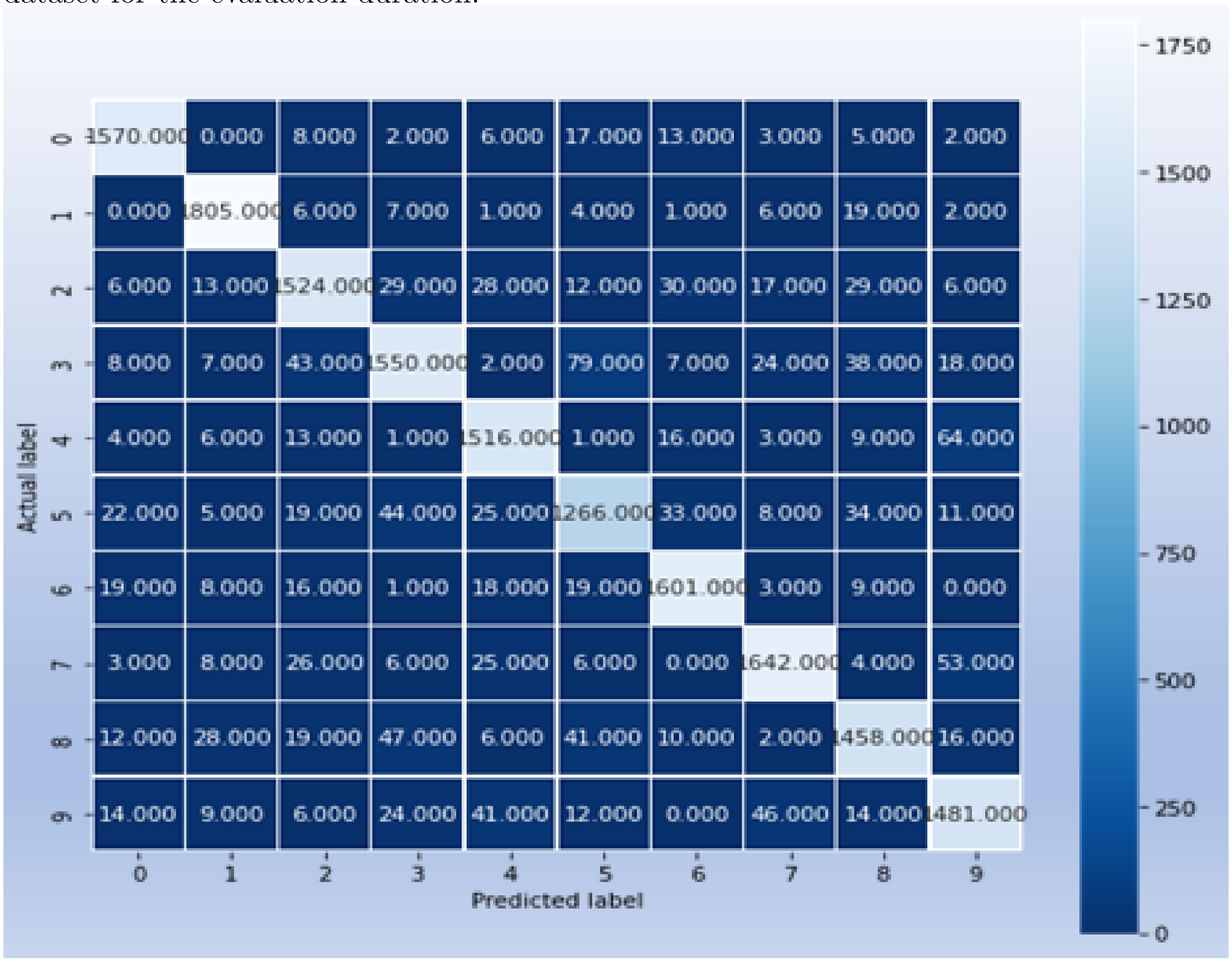


The logistic model trained on the reduced MNIST dataset gives us an accuracy score of 91.74 percent which is a little bit higher than the baseline model accuracy. The confusion matrix is given in Figure 5.6 which shows that the number of incorrect predictions made is a little bit smaller than the baseline model and we got a little higher classification accuracy.

5.2.2 Simple auto-encoders

Simple auto-encoder was applied on the HAR dataset to reduce the data dimensions to 98. For this purpose, the architecture of simple auto-encoders for HAR dataset has been de-

Figure 5.6: Confusion matrix of the logistic model being trained on the reduced MNIST dataset for the evaluation duration.



	0	1	2	3	4	5	6	7	8	9
0	1570.000	0.000	8.000	2.000	6.000	17.000	13.000	3.000	5.000	2.000
1	0.000	1805.000	6.000	7.000	1.000	4.000	1.000	6.000	19.000	2.000
2	6.000	13.000	1524.000	29.000	28.000	12.000	30.000	17.000	29.000	6.000
3	8.000	7.000	43.000	1550.000	2.000	79.000	7.000	24.000	38.000	18.000
4	4.000	6.000	13.000	1.000	1516.000	1.000	16.000	3.000	9.000	64.000
5	22.000	5.000	19.000	44.000	25.000	1266.000	33.000	8.000	34.000	11.000
6	19.000	8.000	16.000	1.000	18.000	19.000	1601.000	3.000	9.000	0.000
7	3.000	8.000	26.000	6.000	25.000	6.000	0.000	1642.000	4.000	53.000
8	12.000	28.000	19.000	47.000	6.000	41.000	10.000	2.000	1458.000	16.000
9	14.000	9.000	6.000	24.000	41.000	12.000	0.000	46.000	14.000	1481.000

signed in a way that the hidden layer compresses the data dimensions to 98. The model is trained for 50 epochs and a batch size of 32. The number of epochs shows how many times the whole data was used for training the model while batch size represents the number of samples that is passed to the neural network at one time. The terms validation loss and training loss here represent the reconstruction loss of the validation dataset and training dataset respectively. It is the check of how well the model has learned from the data. The loss function measures the difference between the original input and the reconstructed input. Loss function depends on the input and output data but mostly mean squared error loss is used as the default loss for training a neural network model. In this study, mean squared error loss has been used as reconstruction error loss.

The curves in Figure 5.7 shows that as the training and validation loss decreases the model is learning more and more from the data. The curves approach towards zero and after 50 epochs it is not getting better more, therefore we have used 50 epochs to train the network.

For MNIST dataset, we make the architecture of simple auto-encoders in a way that the hidden layer reduces the data dimensions to 151. The model is trained for 50 epochs and a batch size of 256.

Figure 5.8 shows that both training and validation loss decreases and approaches towards zero as the model learns more and more from the data. After 50 epochs the curves takes a linear form representing that the model is no longer getting better.

The logistic model trained on the reduced HAR dataset gives us a classification accuracy of 92.62 percent which is approximately equal to the accuracy achieved from the PCA based logistic model.

The logistic model trained on the reduced MNIST dataset gives us a classification accuracy of 91.48 percent, a little bit lower than the baseline accuracy.

Figure 5.9 represent the confusion matrix of the logistic model being trained on the reduced HAR dataset through simple auto-encoders which shows that for the class label 2, 4 and 5 the number of incorrect predictions is greater.

Figure 5.10 shows the confusion matrix for the logistic model trained on the MNIST dataset reduced through simple auto-encoder. The matrix shows that number of incorrect predictions for every class is greater. Therefore, we get a smaller accuracy score.

Figure 5.7: learning behavior of simple auto-encoder for HAR data against the number of epochs

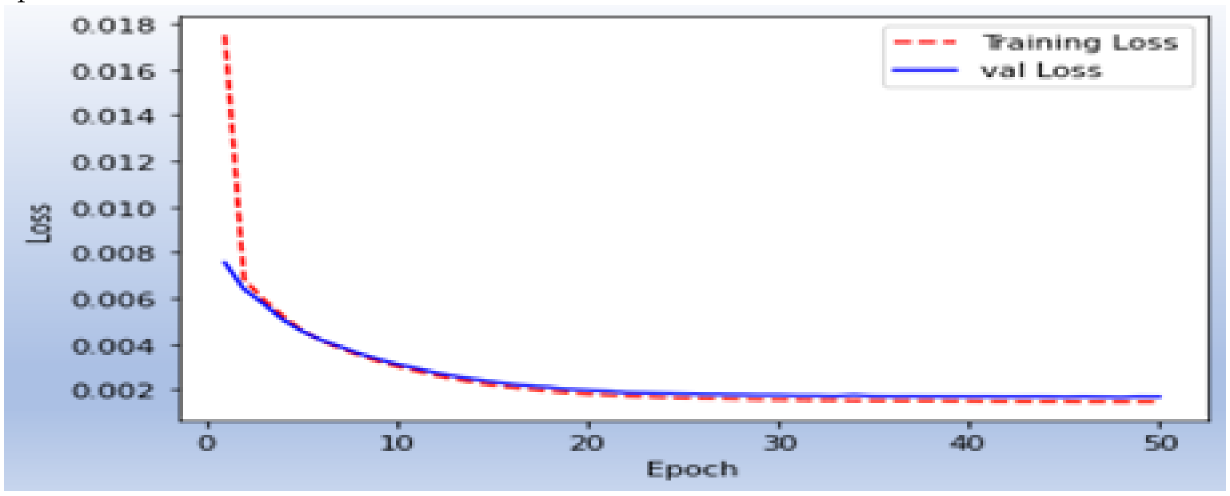


Figure 5.8: learning behavior of simple auto-encoder for MNIST data against the number of epochs

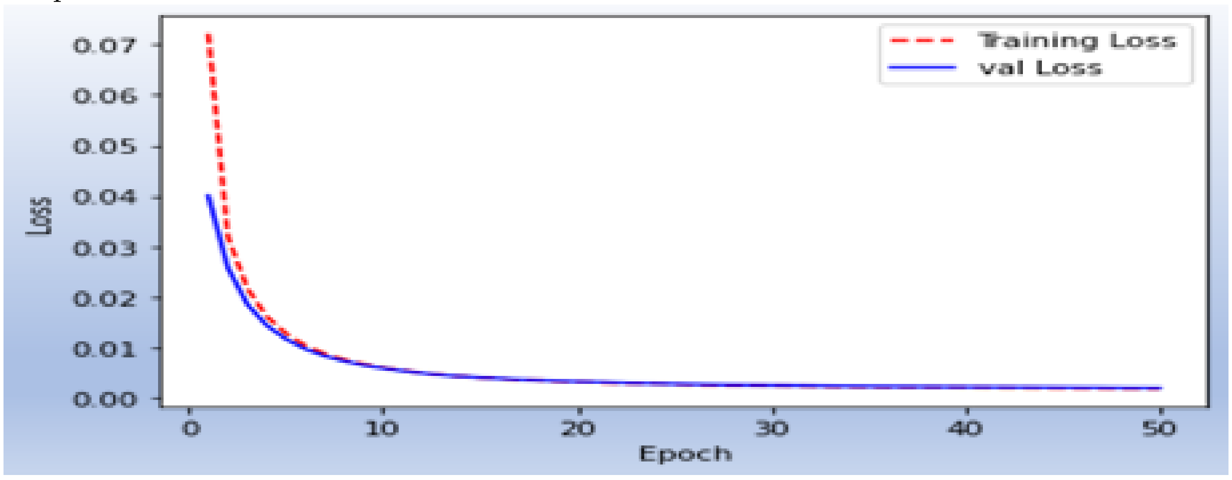
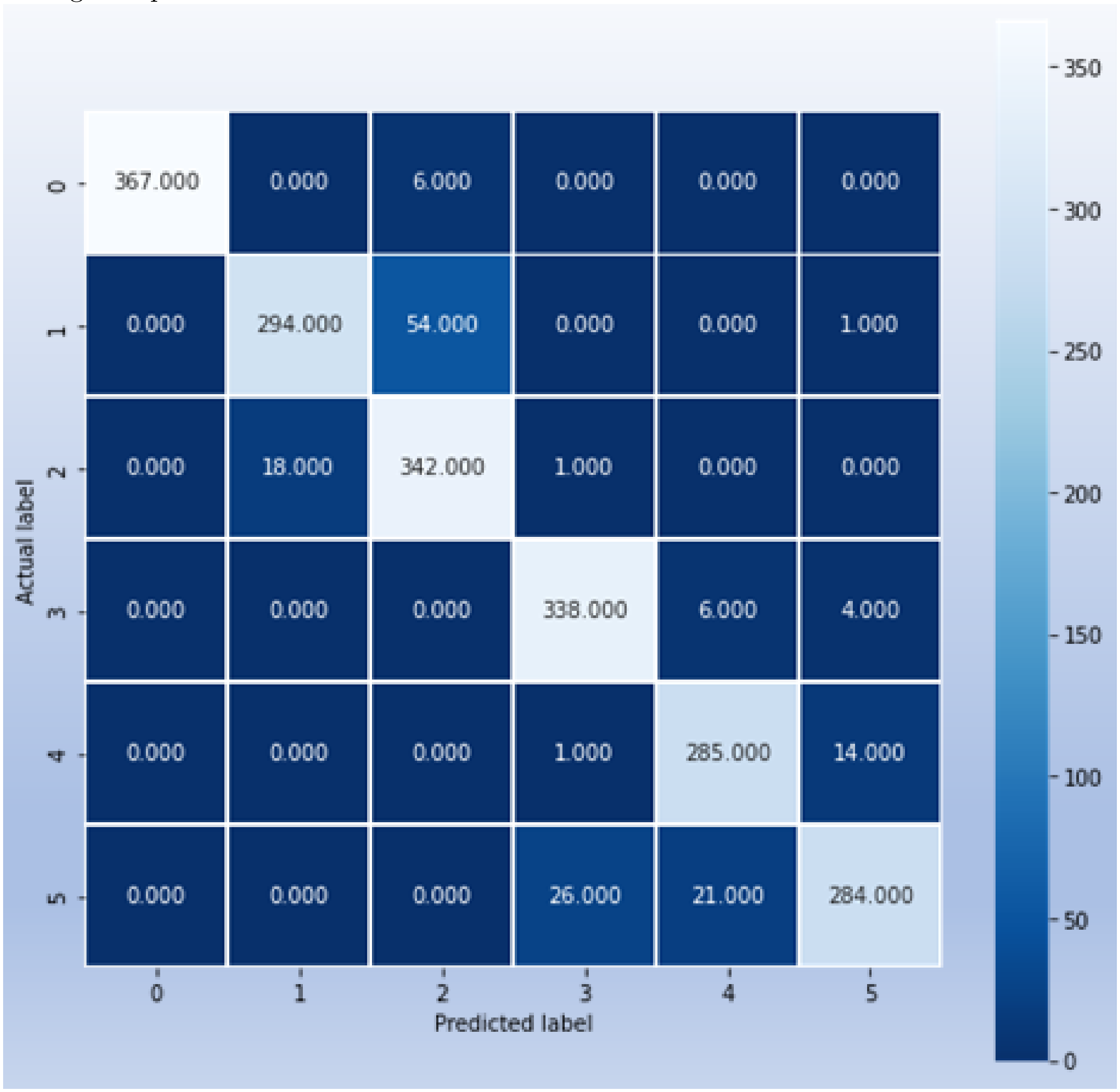
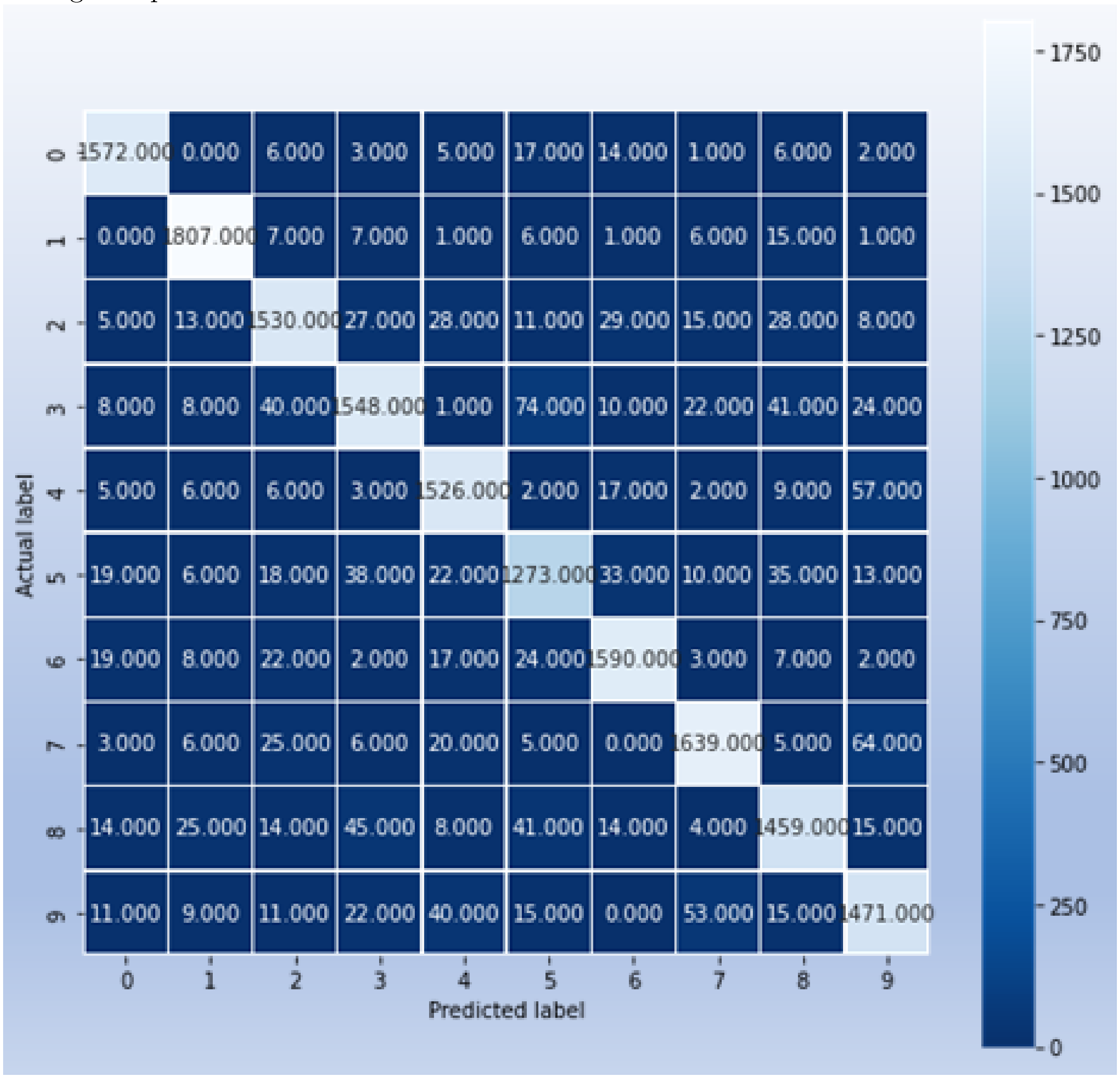


Figure 5.9: Confusion matrix for the logistic model trained on the reduced HAR dataset through simple auto-encoder.



Actual label \ Predicted label	0	1	2	3	4	5
0	367.000	0.000	6.000	0.000	0.000	0.000
1	0.000	294.000	54.000	0.000	0.000	1.000
2	0.000	18.000	342.000	1.000	0.000	0.000
3	0.000	0.000	0.000	338.000	6.000	4.000
4	0.000	0.000	0.000	1.000	285.000	14.000
5	0.000	0.000	0.000	26.000	21.000	284.000

Figure 5.10: Confusion matrix for the logistic model trained on the reduced MNIST dataset through simple auto-encoder.



	0	1	2	3	4	5	6	7	8	9
0	1572.000	0.000	6.000	3.000	5.000	17.000	14.000	1.000	6.000	2.000
1	0.000	1807.000	7.000	7.000	1.000	6.000	1.000	6.000	15.000	1.000
2	5.000	13.000	1530.000	27.000	28.000	11.000	29.000	15.000	28.000	8.000
3	8.000	8.000	40.000	1548.000	1.000	74.000	10.000	22.000	41.000	24.000
4	5.000	6.000	6.000	3.000	1526.000	2.000	17.000	2.000	9.000	57.000
5	19.000	6.000	18.000	38.000	22.000	1273.000	33.000	10.000	35.000	13.000
6	19.000	8.000	22.000	2.000	17.000	24.000	1590.000	3.000	7.000	2.000
7	3.000	6.000	25.000	6.000	20.000	5.000	0.000	1639.000	5.000	64.000
8	14.000	25.000	14.000	45.000	8.000	41.000	14.000	4.000	1459.000	15.000
9	11.000	9.000	11.000	22.000	40.000	15.000	0.000	53.000	15.000	1471.000

5.2.3 Deep-auto encoders

Deep auto-encoders uses many hidden layers stacked upon one another to compress the dataset. Using many hidden layers helps in extracting more complex and hidden features in the dataset. We have used a deep auto-encoders with 4 hidden layers for HAR data to compress the dataset dimensions into the same number of dimensions as reduced by PCA and auto-encoders. We trained the model for 50 epochs and a batch size of 256.
Figure 5.11 represents that the model is not learning in a smooth way and oscillation is present in both the curves. It means that the model is not able to capture the hidden features correctly.

The same 4 hidden layers deep auto-encoders has been used for MNIST dataset and the data dimensions is reduced to 151 latent dimensions. We trained the model for 20 epochs and a batch size of 256.

Figure 5.12 represents that the model is learning in a smooth way and as the number of epochs increases the curves for training and validation loss approaches towards zero. After epoch number 20 the curves become linear and both training and validation loss is not decreasing. Comparing this with the above model for HAR, we see that deep auto-encoders is learning well from the MNIST data. It means that the model can capture the hidden features in MNIST data.

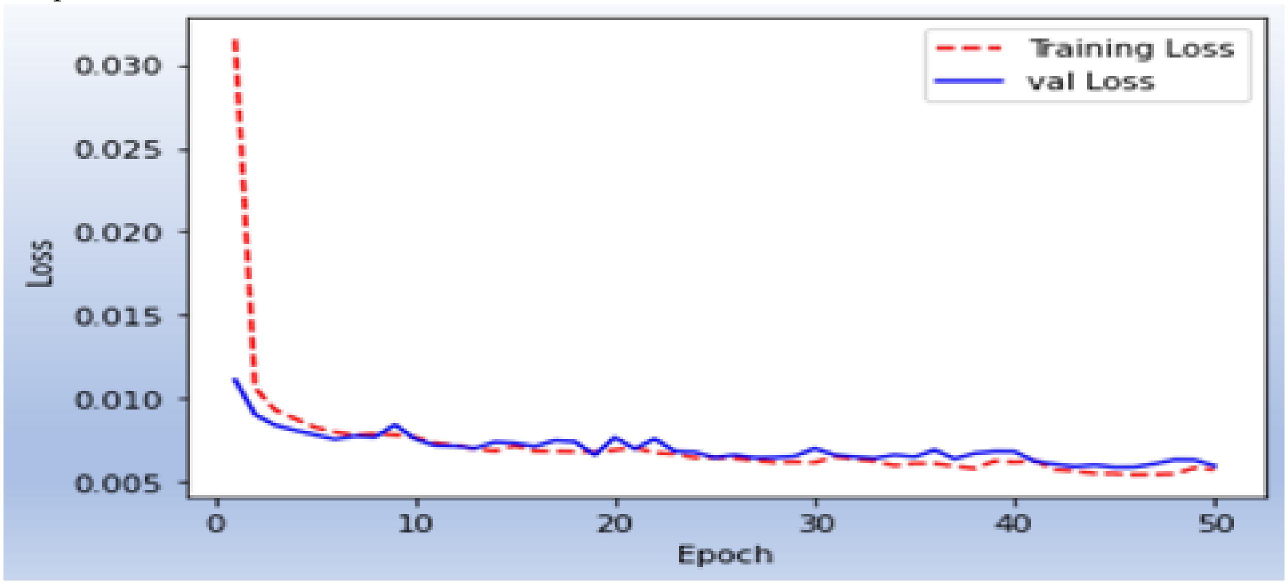
The logistic model trained on the reduced HAR data classified the data with accuracy score of 86.27 percent while for MNIST dataset the logistic model trained on the reduced data gives us a classification accuracy of 93.30 percent which is better than the baseline accuracy for MNIST.

Comparing the results, we can see that for HAR dataset auto-encoders with one hidden layer was able to capture more variation of the data and results in a good comparable accuracy while for MNIST data deep auto-encoders captures the hidden features correctly and classified the data with greater accuracy. It means that for simple and small datasets auto-encoders with a smaller number of hidden layers perform better while for larger and complex datasets deep auto-encoders with more hidden layers is better.

Figure 5.13 represents the confusion matrix of the logistic model which was trained on the reduced HAR dataset through deep auto-encoder. The matrix shows that the number of incorrect predictions is greater than we get by using simple auto-encoder for DR. Therefore, by using deep auto-encoder on HAR, the accuracy of the model was not improved.

The MNIST data was reduced by using deep auto-encoder and the reduced data was feed into the logistic model. The confusion matrix for the model is represented by the Figure 5.14. which shows that for MNIST the deep auto-encoder performance was good as compared to the simple auto-encoder.

Figure 5.11: Learning behavior of deep auto-encoders for HAR data against the number of epochs.



5.3 Evaluation based on execution time

We now compare all the methods based on their time of execution. The execution time is an important measure for evaluating the performance of the model. Table.1 shows the training time of all the models. It is the time taken by the models while learning from the data.

Table 1 shows that for HAR dataset the training time of logistic model was 40 seconds while for MNIST data it was 95 seconds. On both dataset PCA takes very little time of execution in learning from the data. Simple and deep auto-encoders takes a relatively shorter time in learning from the HAR data as compared to learning from the MNIST data.

Figure 5.12: Learning behavior of deep auto-encoder for MNIST data against the number of epochs.

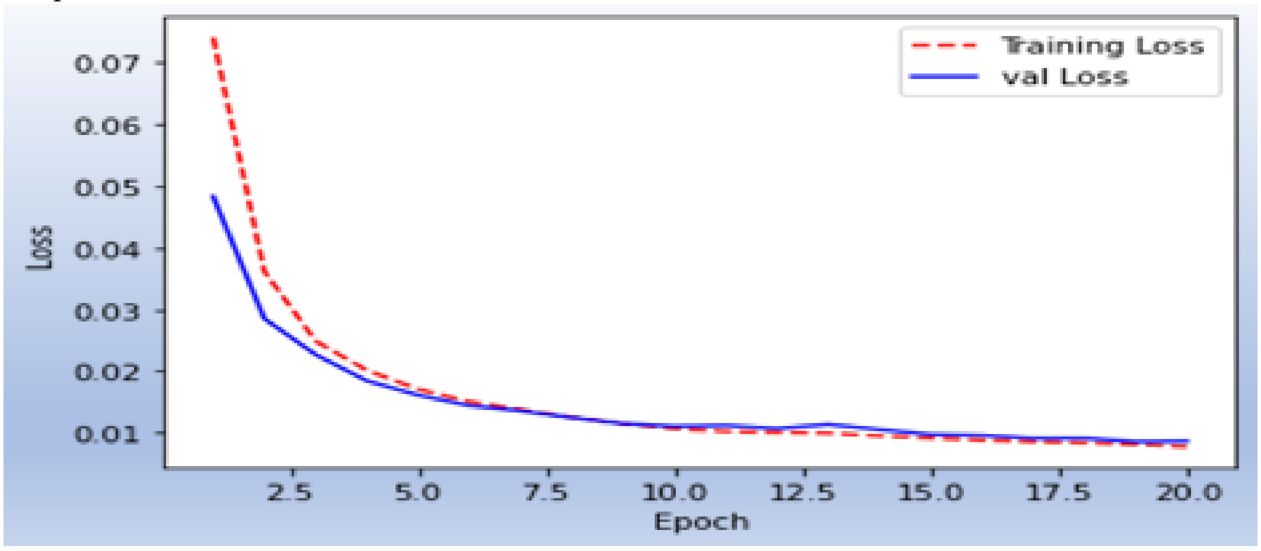


Table 2 shows the training time of the logistic model when it was trained on the reduced datasets after applying PCA and deep auto-encoders.

Table.2 show that for HAR dataset, the logistic model trained on the datasets reduced by PCA and deep auto-encoders takes a training time of 12 and 11 seconds respectively which means that both PCA and deep auto-encoders were able to optimize the training time of the logistic model.

For MNIST dataset, the logistic model trained on the dataset reduced by PCA takes 11 seconds in learning from the data while for simple and deep auto-encoders it takes 335 and 190 seconds respectively which is higher than the time taken by the baseline model. So, for MNIST data only PCA was able to optimize the training time of the logistic model.

Figure 5.13: . Confusion matrix for the logistic model when the model was trained on the HAR dataset reduced by using deep auto-encoder.

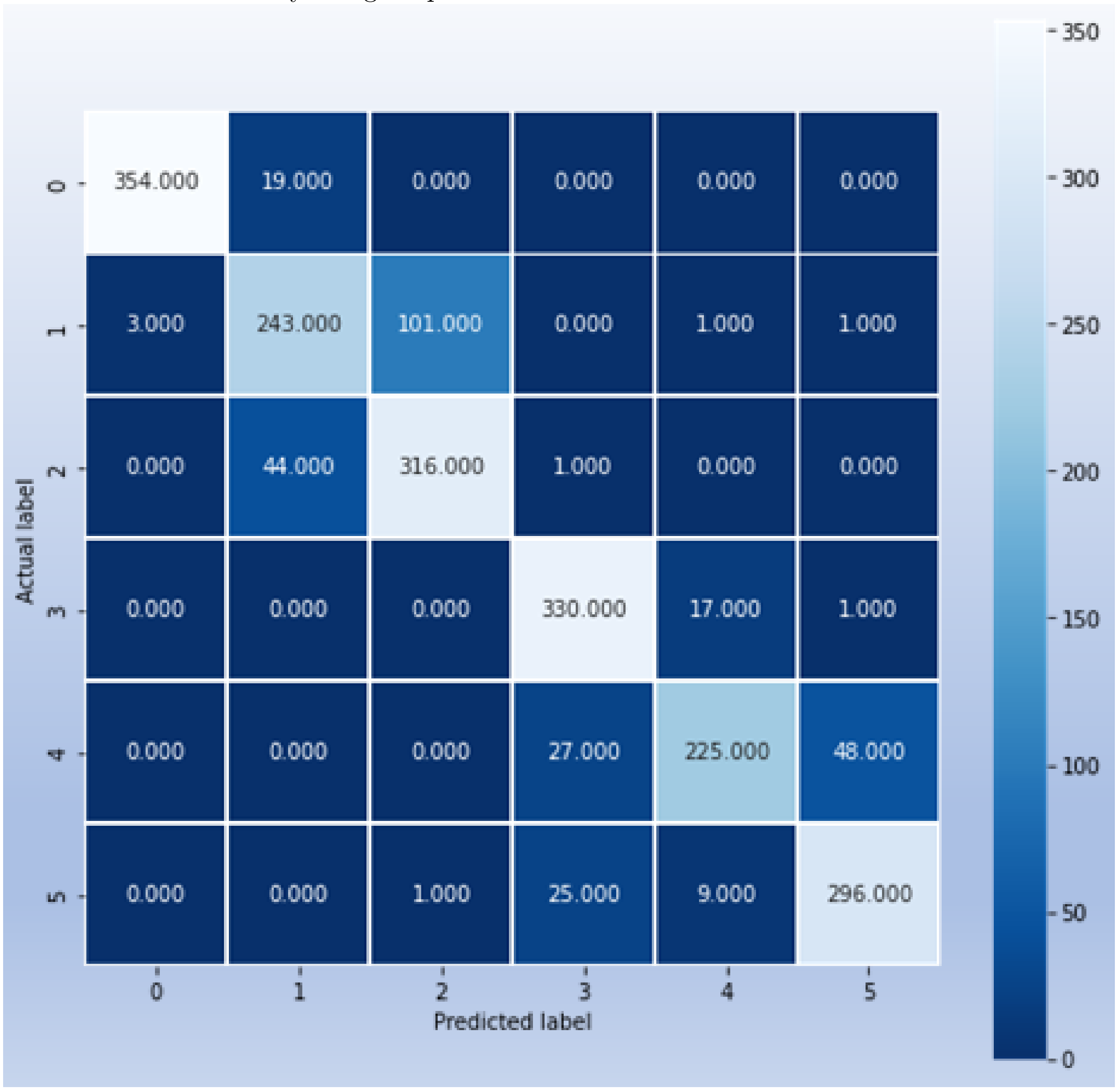
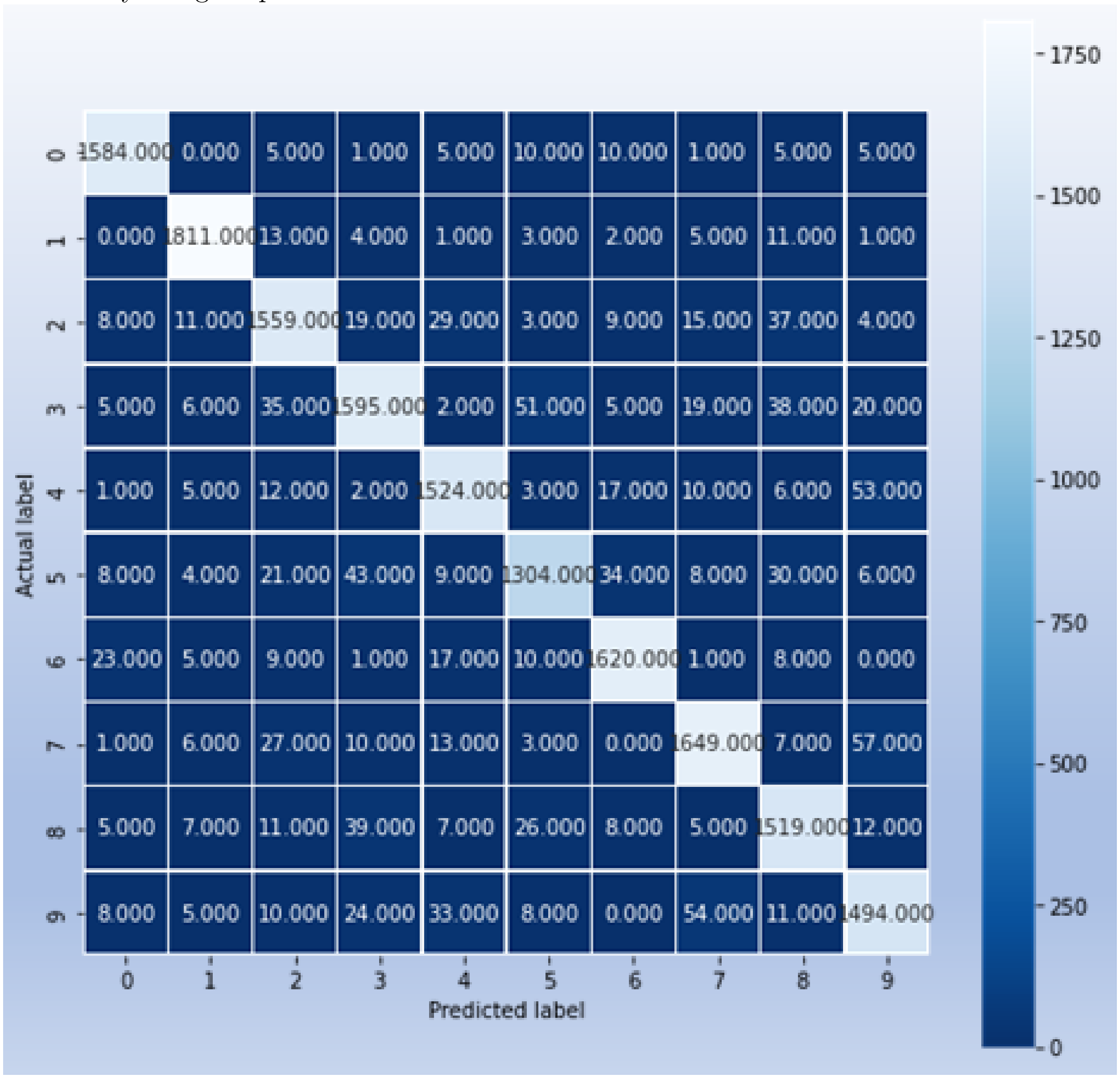


Figure 5.14: Confusion matrix of the logistic model when it was trained on MNIST dataset reduced by using deep auto-encoder.



	0	1	2	3	4	5	6	7	8	9
0	1584.000	0.000	5.000	1.000	5.000	10.000	10.000	1.000	5.000	5.000
1	0.000	1811.000	13.000	4.000	1.000	3.000	2.000	5.000	11.000	1.000
2	8.000	11.000	1559.000	19.000	29.000	3.000	9.000	15.000	37.000	4.000
3	5.000	6.000	35.000	1595.000	2.000	51.000	5.000	19.000	38.000	20.000
4	1.000	5.000	12.000	2.000	1524.000	3.000	17.000	10.000	6.000	53.000
5	8.000	4.000	21.000	43.000	9.000	1304.000	34.000	8.000	30.000	6.000
6	23.000	5.000	9.000	1.000	17.000	10.000	1620.000	1.000	8.000	0.000
7	1.000	6.000	27.000	10.000	13.000	3.000	0.000	1649.000	7.000	57.000
8	5.000	7.000	11.000	39.000	7.000	26.000	8.000	5.000	1519.000	12.000
9	8.000	5.000	10.000	24.000	33.000	8.000	0.000	54.000	11.000	1494.000

Dataset	model type	Execution time in seconds
HAR dataset	baseline logistic regression model	40
	PCA	2
	Simple auto-encoders	41
	Deep auto-encoders	58
MNIST dataset	Baseline logistic regression model	95
	PCA	7
	Simple auto-encoders	80
	Deep auto-encoders	110

Table 5.1: Execution time taken by the methods while being trained on the original data sets

Dataset	model type	Execution time in seconds
HAR dataset	baseline logistic regression model	40
	LR trained on RD by PCA	12
	===Simple auto-encoders	35
	===Deep auto-encoders	11
MNIST dataset	Baseline logistic regression model	95
	LR trained on RD by PCA	11
	====Simple auto-encoders	335
	=====Deep auto-encoders	190

Table 5.2: Time taken by the logistic model when trained on the reduced datasets after applying PCA and auto-encoders

Chapter 6

Conclusion and Discussion

Our comparison of simple and deep auto-encoders with PCA is based on the classification accuracy of the classifier and the time of execution. On HAR dataset, the performance of simple auto-encoder and PCA in terms of the classification accuracy is approximately the same while deep auto-encoders performance is not good enough. But the comparison based on time execution shows that PCA is better than simple and deep auto-encoder both in learning faster from the data and optimizing the execution time of the logistic model.

On MNIST dataset, the performance of deep auto-encoder is better than PCA and simple auto-encoder based on the classification accuracy of the logistic model, but the model training time is longer. PCA is faster than both simple and deep auto-encoder and optimized the execution time of the model.

It is noted that for small and simple types of datasets auto-encoder with a smaller number of hidden layers perform better while for large and complex datasets auto-encoders with many hidden layers outperform other methods.

It is concluded that the performance of PCA on HAR dataset was good both in terms of classification accuracy and optimizing execution's time. On MNIST data, however, PCA does not outperform the deep auto-encoder in terms of the classification accuracy. Simple auto-encoders also achieved a good classification accuracy, but the model was not able to optimize the time of execution. The performance of deep auto-encoders on MNIST dataset was better and it shows that deep auto-encoders performed better on datasets which contain more hidden structures such as images. The results also show that the main issue with the performance of deep learning methods was the training time. We suggest that the performance of deep learning methods may be improved for dimensionality reduction by optimizing the architecture and training time of the model. In future, as more advancements come in the field of deep learning, the performance of auto-encoders may outperform other state of the art dimensionality reduction techniques.

Bibliography

[1] Hervé Abdi and Lynne J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459.

[2] Farzana Anowar, Samira Sadaoui, and Bassant Selim. Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne). *Computer Science Review*, 40:100378, 2021. `doi:https://doi.org/10.1016/j.cosrev.2021.100378`.

[3] Shaeela Ayesha, Muhammad Kashif Hanif, and Ramzan Talib. Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Inf. Fusion*, 59:44–58, 2020.

[4] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[5] L.J. Cao, K.S. Chua, W.K. Chong, H.P. Lee, and Q.M. Gu. A comparison of pca, kpca and ica for dimensionality reduction in support vector machine. *Neurocomputing*,

55(1):321–336, 2003. Support Vector Machines. doi:https://doi.org/10.1016/S0925-2312(03)00433-8.

[6] Barak Chizi and Oded Maimon. *Dimension Reduction and Feature Selection*, pages 83–100. Springer US, Boston, MA, 2010.

[7] Barak Chizi, Lior Rokach, and Oded Maimon. A survey of feature selection techniques. In *Encyclopedia of Data Warehousing and Mining, Second Edition*, pages 1888–1895. IGI Global, 2009.

[8] Quentin Fournier and Daniel Aloise. Empirical comparison between autoencoders and traditional dimensionality reduction methods. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, jun 2019. URL: https://doi.org/10.1109%2Faike.2019.00044, doi:10.1109/aike.2019.00044.

[9] Lianli Gao, Jingkuan Song, Xingyi Liu, Junming Shao, Jiajun Liu, and Jie Shao. Learning in high-dimensional multimedia data: The state of the art, 2017. URL: https://arxiv.org/abs/1707.02683, doi:10.48550/ARXIV.1707.02683.

[10] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. doi:10.1126/science.1127647.

[11] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, 1933.

[12] Rima Houari, Ahcène Bounceur, M-Tahar Kechadi, A-Kamel Tari, and Reinhardt Euler. Dimensionality reduction in data mining: A copula approach. *Expert Systems with Applications*, 64:247–260, 2016.

[13] Xuan Huang, Lei Wu, and Yinsong Ye. A review on dimensionality reduction techniques. *International Journal of Pattern Recognition and Artificial Intelligence*, 33(10):1950017, 2019. `doi:10.1142/S0218001419500174`.

[14] Ian T Jolliffe. Principal component analysis: a beginner's guide—i. introduction and application. *Weather*, 45(10):375–382, 1990.

[15] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016. `doi:10.1098/rsta.2015.0202`.

[16] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. *2014 Science and Information Conference*, pages 372–378, 2014.

[17] Vivek Kumar, Denis Kalitin, and Prayag Tiwari. Unsupervised learning dimensionality reduction algorithm pca for face recognition. 05 2017. `doi:10.1109/CCAA.2017.8229826`.

[18] Joshua Lewis, Laurens van der Maaten, and Virginia de Sa. A behavioral investigation of dimensionality reduction. 01 2012.

[19] Ji Ma and Yuyu Yuan. Dimension reduction of image deep feature using pca. *Journal of Visual Communication and Image Representation*, 63:102578, 2019. URL: `https://www.sciencedirect.com/science/article/pii/S1047320319301932`, `doi:https://doi.org/10.1016/j.jvcir.2019.102578`.

[20] Abdallah Bashir Musa. A comparison of 1-regularizion, pca, kpca and ica for dimensionality reduction in logistic regression. *International Journal of Machine Learning and Cybernetics*, 5:861–873, 2014.

[21] Hadeel S. Obaid, Saad Ahmed Dheyab, and Sana Sabah Sabry. The impact of data pre-processing techniques and dimensionality reduction on the accuracy of machine learning. In *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*, pages 279–283, 2019. `doi:10.1109/IEMECONX.2019.8877011`.

[22] G. Thippa Reddy, M. Praveen Kumar Reddy, Kuruva Lakshmanna, Rajesh Kaluri, Dharmendra Singh Rajput, Gautam Srivastava, and Thar Baker. Analysis of dimensionality reduction techniques on big data. *IEEE Access*, 8:54776–54788, 2020. `doi:10.1109/ACCESS.2020.2980942`.

[23] R. Rianto, Achmad Mutiara, Eri Prasetyo, and Paulus Santosa. Improving the accuracy of text classification using stemming method, a case of informal indonesian conversation, 07 2020. `doi:10.21203/rs.3.rs-41431/v1`.

[24] Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Auto-encoder bottleneck features using deep belief networks. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4153–4156. IEEE, 2012.

[25] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *MLSDA'14*, 2014.

[26] Nema Salem and Sahar Hussein. Data dimensional reduction and principal components analysis. *Procedia Computer Science*, 163:292–299, 2019. 16th Learning and Technology Conference 2019Artificial Intelligence and Machine Learning: Embedding the Intelligence. `doi:https://doi.org/10.1016/j.procs.2019.12.111`.

[27] Basna Mohammed Salih Hasan and Adnan Mohsin Abdulazeez. A review of principal component analysis algorithm for dimensionality reduction. *Journal of Soft Computing and Data Mining*, 2(1):20–30, Apr. 2021. URL: `https://publisher.uthm.edu.my/ojs/index.php/jscdm/article/view/8032`.

[28] C. O. S. Sorzano, J. Vargas, and A. Pascual Montano. A survey of dimensionality reduction techniques, 2014. URL: `https://arxiv.org/abs/1403.2877`, `doi:10.48550/ARXIV.1403.2877`.

[29] Carlos Oscar Sánchez Sorzano, Javier Vargas, and Alberto D. Pascual-Montano. A survey of dimensionality reduction techniques. *ArXiv*, abs/1403.2877, 2014.

[30] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[31] Laurens van der Maaten, Eric O. Postma, and Jaap van den Herik. Dimensionality reduction: A comparative review. 2009.

[32] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9(nov):2579–2605, 2008. Pagination: 27.

[33] Tomas Vantuch, Vaclav Snasel, and Ivan Zelinka. Dimensionality reduction method's comparison based on statistical dependencies. *Procedia Computer Science*, 83:1025–1031, 2016. The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016) / The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016) / Affiliated Workshops. URL: `https://www.sciencedirect.com/science/article/pii/S1877050916302514`, `doi:https://doi.org/10.1016/j.procs.2016.04.218`.

[34] Vishwa Vinay, Ingemar J. Cox, Kenneth R. Wood, and Natasa Milic-Frayling. A comparison of dimensionality reduction techniques for text retrieval. *Fourth International Conference on Machine Learning and Applications (ICMLA'05)*, pages 6 pp.–, 2005.

[35] Sumithra V.S and Subu Surendran. A review of various linear and non linear dimensionality reduction techniques. 2015.

[36] J. Wang, Haibo He, and Danil V. Prokhorov. A folded neural network autoencoder for dimensionality reduction. In *INNS-WC*, 2012.

[37] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 496–503, 2014.

[38] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

[39] Killan Q. Weinberger and Lawrence K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. 2006.

[40] Haozhe Xie, Jie Li, Qiaosheng Zhang, and Yadong Wang. Comparison among dimensionality reduction techniques based on random projection for cancer classification. *Computational biology and chemistry*, 65:165–172, 2016.

[41] Rizgar Ramadhan Zebari, Adnan Mohsin Abdulazeez, Diyar Qader Zeebaree, Dilovan Asaad Zebari, and Jwan Najeeb Saeed. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. 2020.

[42] Jun-Hai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. Autoencoder and its various variants. *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 415–419, 2018.