Feature Extraction using Autoencoders for Various Challenging Tasks of Pattern Recognition

Submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy in Computer Science

by

Debasrita Chakraborty

Under the supervision of
Prof. Ashish Ghosh

Machine Intelligence Unit
Indian Statistical Institute
203 B.T. Road, Kolkata. India - 700108
December, 2022

To my son, Samadarshi

You inspired me to be the one worthy of admiration and my life's aim will be to never let you down.

Acknowledgement

Without the encouragement and guidance of a certain number of great people, this thesis wouldn't have been possible. I extend my gratitude to all of them for making this experience possible. My heartfelt thanks go to Professor Ashish Ghosh, my supervisor who has always been a fatherly figure to me. This work would have hardly finished without his passion, inspiration, guidance, and constant support when I was juggling my way through academics and motherhood simultaneously. Special mention needs to be made of Professor Susmita Ghosh who always made time for my academic needs whenever I needed her.

I would also like to thank our Director Prof. Sanghamitra Bandyopadhyay, our current Dean of Studies Prof. Debasis Sengupta, former Dean of Studies Prof. Pradipta Bhattacharya, and Goutam Mukherjee, Convener of Ph.D. D.Sc. committee Prof. Pradipta Majhi and my mentors during the coursework for their technical support on my study.

I would like to thank all my friends and seniors in Machine Intelligence Unit especially, my colleagues Anwesha Law, Subhadip Boral, Debayan Goswami and my senior Rahul Roy. I would like to express my gratitude for all other study collaborators' shared know-how and perspectives and their useful input.

Finally, I thank my family deeply and sincerely for their unceasing and unparalleled love, support, and assistance. I am thankful to my brother Debayan whose advice pushed me to pursue this degree. I am grateful to my parents Molay Chakraborty and Dalia Chakraborty who encouraged me selflessly to discover new ways to live and to pursue my destiny. I would like to thank my husband Samir for always showing how proud he is of me. Finally, my thanks go to my little one Samadarshi for showing me the wonder of life amid the difficult moments of preparing this thesis.

Place: Indian Statistical Institute, Kolkata
Date: 20 July, 2022 Debasrita Chakraborty

Abstract

Feature extraction is a technique through which existing features are transformed into a different (usually smaller) dimension. This conceptually means that the data is represented in a different aspect than the original one. This kind of data representation is among the key machine learning principles and often helps in finding some interesting relationships in the data. Detecting the structure and automated identification of patterns in datasets is indeed a suitable benefit as it facilitates understanding of the process described by the data. Hence, the effectiveness of machine learning algorithms vastly depends on the types of features they rely on due to the multi-dimensionality of information that feeds the model. Depending on how data is represented, different models may view the problem in different ways and try to solve it using unique techniques. Different pattern recognition issues and challenges of classification have been improved by deep neural networks in recent years due to their inherent capability of learning from raw data. Deep architectures have also demonstrated their efficiency in recording latent data representation characteristics.

Even though deep neural networks are well capable of handling complex data, the challenges posed by imbalanced datasets, high dimensional datasets, highly chaotic time-varying datasets, or decentralised datasets are difficult to handle. Therefore, the main focus is on four such situations of complex datasets where standard deep neural networks fail. However, using an autoencoder and combining it with other techniques has proven to be beneficial in such conditions. In this thesis, the efficacy of autoencoders is argued in some really interesting areas. The investigations show that autoencoders are particularly useful for datasets where there is an imbalance, lack or absence of labelled samples and chaos. Thus, the successive chapters look into some application cases of the above scenarios and explore how the autoencoder supplemented methods deal with the challenges in those applications.

For the first task, a straightforward outlier detection problem is handled. It is seen that the autoencoders are very well capable of enhancing the performance of outlier detection models. So, the problem is extended to another use case where the data has somewhat of a local imbalance, high complexity and high dimensionality. This is observed for remotely sensed hyperspectral images where the task is to detect changes between such a pair of co-registered bi-temporal images. The tasks undertake cases where the label information is partially absent and completely absent respectively. It is observed that autoencoders are well suited to capture the changing neighborhood information surrounding the same pixel location of the two images. Autoencoders are also examined under conditions where the data is unpredictable as seen for OHLC (open high low close) stock prices. It is seen that transformation by autoencoders are much more informative than the original feature space. This is why an autoencoder supplemented prediction model helps in making better predictions about the future OHLC stock prices. Since the above methods are fairly cases of a centralised data setting, it was also necessary to examine how the autoencoders fair for decentralised imbalance. Thus, the efficacy of autoencoder is inspected under federated learning. It is seen that pre-training by autoencoders is particularly useful when the data is imbalanced and thus can be used for situations where the data distribution among the local nodes is non-i.i.d.

Since autoencoders are unsupervised feature extractors, they do not incorporate any kind of class information during the training process. The study investigates if the use of such training of autoencoders lead to a competitive edge in performance.

Contents

List of Figures

List of Tables

Abbreviations

AE	Autoencoder
AI	Artificial Intelligence
ASL	American Sign Language
BM	Boltzmann Machine
BP	Backpropagation
BTST	Buy-Today Sell-Tomorrow
CAE	Convolutional Autoencoder
CNN	Convolutional Neural Network
DAE	Deep Autoencoder
DBN	Deep Belief Network
DL	Deep Learning
DNN	Deep Neural Network
DR	Detection Rate
EA	Evolutionary Algorithm
ELM	Extreme Learning Machine
FL	Federated Learning
FLCAE	Functional Link Convolutional Autoencoder
FNR	False Negative Rate
FPR	False Positive Rate
GAN	Generative Adversarial Network
GRP	Gaussian Random Projection
GRU	Gated Recurrent Unit
HSI	Hyperspectral Image

i-Forest	Isolation Forest
i.i.d	Independent and Identically Distributed
KDA	Kernel Discriminant Analysis
k-NN	*k*- Nearest Neighbours
k-PCA	Kernel Principal Components Analysis
LLE	Locally Linear Embedding
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MAP	Maximum a Posteriori Probability
MAPE	Mean Absolute Percentage Error
MLNN	Multi-layer Neural Network
NN	Neural Networks
OHLC	Open High Low Close
PCA	Principal Components Analysis
pdf	Probability Density Function
PNN	Probabilistic Neural Networks
PR	Pattern Recognition
RBF	Radial Basis Function
RBM	Restricted Boltzmann Machine
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RP	Random Projection
RSI	Remotely Sensed Image
SAE	Stacked Autoencoder
SVDD	Support Vector Data Description
SVM	Support Vector Machine
TNR	True Negative Rate
t-SNE	t-Distributed Stochastic Neighbor Embedding
VAE	Variational Autoencoder

Chapter 1

Introduction

1.1 Deep Neural Networks

A well-trained multi-layered neural network (MLNN) could learn many complex decision regions. It was proved [1, 2] that there always exists a three-layer network (with some desired numbers of neurons) that can approximate an arbitrary nonlinear, continuous, multi-dimensional function f with any desired accuracy. However, the number of neurons could go up to some thousands for a high-dimensional polynomial function. It was proved that if a circuit with depth d needed $\mathcal{O}(n)$ nodes to model a function, one with depth $d-1$ will need $\mathcal{O}(2^n)$ nodes [3, 4]. Intuitively, a four-layer neural network would perform better in modelling non-linear problems with effectively lesser neurons than a three-layer network.

However, many researchers found that having more than 2 hidden layers provided only diminishing effects (slowed the training time significantly but did not improve precision). This is due to the problem of vanishing or exploding gradients. Researchers have avoided these problems by pre-training, savvy initialisation (like Xavier Glorot [5]) or other modifications (like using ReLU activation) and have finally managed to make deep learning

(DL) successful in many aspects. The architecture of the network, amount of training data, loss function, and regularization all play crucial roles. As a rule of thumb, one needs $\mathcal{O}(d)$ examples, where d are the degrees of freedom (i.e, total number of weights and bias parameters in the network) [6]. It means that more data always beats wiser algorithms. As the data complexity grows, shallow models fail to give good accuracy and deep neural networks (DNNs) come into the picture.

The concept of DL alleviates problems in which shallow learning (having less than two non-linear transformations) is affected by the curse of dimensionality [7]. DL involves multiple layers of non-linear transformations along the successive layers of the network. In doing so, a set of statistically robust features is automatically extracted from the data. DL is a fairly new addition to the literature on Machine Learning (ML) research. Using DL, the prime objective of ML, which is to build a reliable Artificial Intelligence (AI), seems plausible now. DL research is concentrated around the use of MLNNs, which are renamed as DNNs because they have better training tricks now. Currently, many variations of DNNs exist in the literature. Autoencoders (AEs) [8, 9], Deep Belief Networks (DBNs) [10], Convolutional Neural Networks (CNNs) [11], Recurrent Neural Networks (RNNs) [12] all have specific purposes and excel at their respective application areas. The beauty of DNNs lies in their feature extracting abilities across multiple layers.

Deep networks like Autoencoders (AEs), Convolutional Neural Networks (CNNs), etc have recently gained popularity as excellent feature extractors. The process of training a network to encode the features and then again decode them back from the encoded version makes them learn efficient feature representations in the encoded dimension. However, these feature representations are problem-specific and depend on the type of dataset provided to them. Multi-layer neural networks have been in the research field for a long time. However, the sudden popularity of DL grew because of the ability to train DNNs. One of the ways to train a deep network is to use AE pre-initialisation strategy. It is often seen as a transfer learning scenario too and is claimed to be the most used technique for weight initialisation. This kind of pre-initialisation technique has also solved the age-old difficulty in training an MLNN using backpropagation [8, 13]. The problems of vanishing and exploding gradients [14, 15] are handled in such a pre-initialisation strategy. It has been argued [9, 12, 16] that such a pre-initialisation strategy act as a regularisation to the network and sets the weights of the network to a close enough approximate value. Re-training the network only means a fine-tuning of the weight values. This beauty of pre-training has therefore surpassed the concern for increased time complexity in a two-phase backpropagation process. Nevertheless, it should not divert us from the fact that in order to make a network learn, we are dedicating twice the amount of time for training. Anyway, since the testing time for neural networks (NNs) is negligible, a DNN is preferred for its enhanced performance. So, the pitfall of the training time complexity is overlooked. However, as DNNs are NNs, there are other issues that they suffer from.

1.2 Problems Identified

DNNs like AEs or CNNs being an extension of the standard NNs suffer from masking and swamping effects of imbalanced data [17]. This situation is aggravated as the imbalance becomes severe. Anomalies or outliers are unusual occurrences and it is exceedingly difficult to accurately describe them. They are an extreme case of imbalanced data. When outliers or anomalies are seen in a dataset, they often reveal some new physics behind their occurrences. Detecting such anomalies is not an easy task. The phenomenon of marking typical events as abnormalities are called swamping. In presence of anomalies, the majority (non-anomalous) often biases the algorithm towards them. Thus, the anomalous data samples originally belonging to a separate cluster or class are combined into a single cluster or class when clustering or classification techniques are employed respectively. This causes the anomaly samples to be identified as regular data points. In such situations, the characteristics of the outliers are not identified in principle. This phenomenon is referred to as masking. Swamping and masking effects are much more likely if the size of the dataset is big. As DNNs need a huge amount of data to learn, they are much more prone to these effects. Thus, a system needs to be developed which can leverage the power of AE-based feature extraction as well as reduce the swamping and masking effects. This way we may find interesting physics behind the occurrence of the outlier data which are not known otherwise.

While talking about the concerns of revealing new physics, the idea of finding changes is quite intriguing. The above scenario described the issues with DNNs when the data was labelled as 'normal' and 'outlier'. Some samples may gradually deviate from their 'normal' behaviour over a fair span of time. If observed within a short time, they seem the same, but after a fairly good amount of time, the changes are prominent. This is particularly seen when there are gradual changes over a place. If a satellite captures the photograph of the same location over a certain span of time, it may be seen that the images are different in comparison to each other suggesting changes. The prominent changes might be visible, but the tiny ones are often overlooked by most of the change detection models as they are surrounded by a large unchanged area. Keeping a track of those subtle changes is crucial to know about the changing dynamics of the place. The problem becomes even more complex if the data is unlabelled. Usually studying these changes is completely driven by RSIs captured by satellites. Detecting changes is not an easy task as the images have comparatively low resolution and one pixel on the image may represent a larger area (a few meters to kilometres). So, a change that has occurred in a small area is difficult to detect. The problem becomes even more complex if the dimensionality of the images increases as in hyperspectral images (HSIs). The total number of changed pixels is often far less than the number of unchanged pixels. Moreover, for the tiny changes, the local neighbourhood of the changed pixels is overwhelmed with unchanged ones. Even though the imbalance is not as pronounced as that of an outlier dataset, the local neighbourhood imbalance often makes the model biased. So we can say that this problem is mostly related to detecting anomalies, except that it is looking for a change or difference instead of finding an abnormality. In this case, the HSI obtained on a former date is considered as the normal perspective of the location photographed and the HSI obtained on a later date may be considered as the changed or abnormal perspective. Any pixel that represents a change needs to be detected. However, these images are inherently complex and noisy. In this situation, noise may also be perceived as a change without the presence of human annotations. Moreover, tiny subtle changes are often overpowered by the surrounding unchanged pixels. In this regard, the potential of AEs to extract meaningful features needs to be explored.

While we are concerned with the issue of changes over time, there is another highly relevant issue that may arise when the changes happen rapidly over a very short duration of time. This requires a time-series analysis of the occurrences. When it comes to time-series data analysis, the data may adopt a chaotic nature along with the complexities stated above. For the two problems discussed above, we were more concerned about finding the anomaly or change. However, while we scrutinize time-varying datasets, we are trying to find order in chaos. We do not have any idea about the underlying reason for the behaviour of the dataset and it may seem chaotic, but according to chaos theory, we may attempt to view this uncertainty as an effect of some latent cause. One such application area is predicting the fluctuations of the financial markets. When it comes to research, it has been an extremely difficult assignment given its inherent complexity, dynamics, and chaos. This makes it exceedingly difficult to forecast the behaviour of stock market values in the future. The behaviour is highly uncertain and there is no concept of 'normal' behaviour in such time-series trends. The trends fluctuate very rapidly depending on numerous factors. Since the behaviour of stock prices is mostly dependent on the country's economic scenario, it becomes rather difficult to predict whether a sudden change will take place or not. Thus, a reliable forecasting model needs to be proposed which can be supplemented by AE-based latent feature extraction.

Since the preceding problems concentrated on a centralised data setting, attention must also be extended to decentralised data settings. The training requirements of DNNs demand a large amount of data for training and the data is often found in a decentralised manner. If the data is globally imbalanced (e.g. outlier detection datasets) and is distributed across multiple local devices in a decentralised manner, the local chunks of data do not follow the simplified assumptions of traditional federated learning (FL). As the data becomes imbalanced in each local server, the traditional FL algorithms to train DNNs face problems with degrading performance. The problem is even more pronounced if most of the local datasets in one or more local devices contain samples from only the majority class. With imbalanced datasets, therefore, the chance of a local dataset having no samples from the minority class (classes) increases as the number of local devices increases. So, a proper method to deal with the stated problem needs to be proposed. In this regard, AE-based pre-training of local DNN models in each local device must also be given importance as AEs can efficiently find suitable features for the class it observes the most. So, one local model which is trained on a local dataset having one class may be important to detect the majority class samples, and the other local model which is trained on a local dataset having some minority class samples may be given more importance for detecting the minority class samples.

In this thesis, the above problems of pattern recognition (PR) have been selected and studied in conjunction with the strengths of AEs. The thesis explores four challenging scenarios of datasets namely, imbalanced data, high dimensional data, chaotic time-varying data, and distributed decentralised data. The research looks into real-life cases of the above scenarios and proposed methods to deal with the relevance of AEs in those applications.

1.3 Motivation

Typical usage of an NN is in a supervised training scenario. The training includes a sample with a label at the output. The NN attempts to learn how to map the input into the specified output label. However, when the output label is substituted with the input vector itself, the network will try to discover the mapping from the input to the input itself. This would mean learning a function of identity which is trivial. However, if the NN is built in such a way that it cannot simply duplicate the input, then only the most important details about the data will be captured by the network. This restriction offered up an unexplored area of NN applications which became popular as AEs [8, 9]. Chapter 2 gives a detailed description of this particular network and its recent advancements in literature. Usually, the concept of Ae has become synonymous with the study of DL. Almost all ML algorithms rely heavily on the quality of the data it is trained on. The results and the performance are based upon the features used to construct the training model. Earlier, the ML community tried to improve the quality of the data using efficient feature engineering techniques. However, in the process of general AI, algorithms must be less reliant on the quality of data and learn to generalise the descriptive aspects of the data on their own. AEs are one of the tools in DL which has the ability to extract meaningful features from the data on its own. The work described in the thesis concerns designing some efficient algorithms using AEs for learning from imbalanced, high dimensional, time-variant, and decentralized data respectively.

The thesis mostly revolves around the beauty of AEs and their efficient feature representations. Often, these representations are termed as a 'black box' [18] and are inexplicable. Even though DNNs are well capable of handling complex data, the challenges posed by imbalanced datasets, highly chaotic time-varying datasets or decentralised datasets are difficult to handle. The thesis mainly concerns four such situations of complex datasets where a standard DNN fails. However, using an AE and combining it with other techniques has proven to be beneficial in such conditions. The increased popularity of DL stems from the notion of representational learning, where the model finds representations necessary to extract the characteristics from data automatically. DL's core idea is to pass the data through several layers where a new representation is created at every level by using certain nonlinear functions on the output of the previous level. In recent years research has demonstrated that this multilayered hierarchical learning may lead to unparalleled outcomes. The AE is a fantastic variant of an NN where each data point(sample) is mapped via several layers of hidden representations. This map is irrespective of basic knowledge about the truth of the matter, i.e. the network requires no information about class or category labels. Training an AE often includes reducing the error of reconstruction which helps in learning important hidden representations.

Since AEs are unsupervised feature extractors [19], they do not incorporate any kind of class information during the training process. The interesting question in this context is whether AEs are capable of learning class-specific features. It has been shown [16] that AEs tend to map similar data to a similar range. Thus, they are capable of efficiently capturing the class-specific data manifold while eliminating redundancy.

Like principal component analysis (PCA), AEs can be a way to learn the latent representations in lower dimensions. However, unlike PCA, the features are not transformed linearly. The version of kernel PCA (k-PCA) operation is somewhat identical to the workings of an AE [20], but AEs are a more generalised concept. The method of extracting features using an AE seems to be useful, particularly if a problem is defined by complicated interrelationships of variables. An AE is simply trained to reconstruct the input data by passing it through one (or more) non-linear hidden layers. It consists of two main parts namely, an encoder that encodes the input data and a decoder that decodes the encoded data to reconstruct the input back.

This transformation in the encoder and corresponding reconstruction in the decoder would have been an extremely tough operation if the input features behaved independently. Nevertheless, if there is any latent structure in the data which describes the relationship between the features, it is possible to learn and thus harness this structure while pushing the input via the middle (or bottleneck) layer of the network. The optimal AE model optimizes between its sensitivity to accurately reconstruct the input and its generalising capacity so as not to overfit and memorise the data. This optimisation obliges the model to retain just the right amount of information necessary to rebuild the input without any redundancy.

The crucial question that arises now is if an AE would be capable enough to extract the right amount of information in case of imbalanced, time-variant, and decentralized data respectively. This has motivated us to look into the feature extracting capabilities of AEs in various such challenging PR problems.

1.4 Contributions

The main contribution of the thesis is to establish that an AE is an extremely versatile feature extraction tool in ML. The thesis focuses on demonstrating that the compactness and the adaptability offered by an AE are beneficial in any challenging pattern recognition task. The study presented in the thesis has brought out many interesting aspects about an AE. It has shown how an AE can be especially useful when the data is imbalanced. Moreover, an AE is also effective at bringing out some relevant infrequent features (along with commonly observed ones) when used in an unsupervised manner. Retraining the AE-enabled network with labels merely fine-tuned the classifier. AE is also shown to be useful for chaotic datasets where it can capture the underlying relationships within the data. The thesis concerns four different challenges. It aims to investigate the potency of using AEs in cases where the dataset poses some inherent obstacles in training a DNN. The novelty of the thesis pertains to exploring the applicability of the feature representation learnt by an autoencoder where a standard DNN would fail or give a mediocre performance. This kind of study has not been done in the literature and AEs are often replaced with better alternatives. We argue that with slight alterations in their structure as well as their objective function, AEs can be the core of a possible solution to many problems.

Diversity in the challenging scenarios required different autoencoder architectures and hence cross-comparison of the proposed models is impractical. However, in all the methods, the main contribution of the autoencoder is highlighted and it is also shown how the models would perform if the AE is replaced by some other suitable feature extraction method or is completely omitted. This thesis explores the suitability of AEs for problems where repetitive or chaotic patterns often overwhelm a classifier and in many cases bias the model towards the majority of the model fails to make an inference respectively.

The first contribution of this thesis involves a straightforward scenario of supervised outlier detection. DNNs being an extension of multi-layer neural networks suffer a lot from the masking and swamping effects of outliers. The problem of outlier detection may also be viewed in a different fashion where the data can have multiple types of outliers. Capturing such a variety in outliers would be useful in discovering different types of anomalies and understanding the physics behind their origin. We have proposed such a scenario to be treated in an entirely different light as a multiple outlier type detection problem. This direction has not been explored in the literature earlier. We argue that there is a need for a unified framework which can detect multiple types of anomalies in a dataset. Thus, a novel framework has been developed and investigated to detect multiple outlier types inspired by the projection capabilities of AEs. Surprisingly, the use of AEs improved the quality of outlier detection significantly, which validates our motivation for using an AE in the first place.

Since AEs are better at learning the patterns, they can be extended to image datasets as well. However, for that, we would need Convolutional Autoencoders (CAEs) which are much more suitable for images. The second objective of the thesis is to study the efficacy of CAEs in image datasets. As a preliminary experiment, the CAEs have been used to develop a trigger detection system using the 24 static hand gestures of the American Sign Language (ASL). This was extended to constructing a customizable switch that can turn 'on' if it finds a trigger gesture in any video that it receives and stays 'off' if it does not. It is seen that CAE is an efficient feature extractor and may be used for images with higher dimensions. This is observed in the case of hyperspectral images. To see the competence of a CAE, a practical situation of the unlabelled or partially labelled imbalanced dataset is chosen. The problem of binary change detection may be regarded as an imbalanced dataset because the changed pixels are always lesser in proportion to the unchanged pixels. This causes a huge problem for any standard change detection techniques as the model mostly comes across unchanged data and the minute changes are often overlooked as noise. Handling hyperspectral images is a challenging task due to a large number of spectral bands present in the data. Researchers, therefore, try to handle it by reducing dimensions. Moreover, manual labelling of changes in a pair of remotely sensed hyperspectral images is costly and time-consuming. As the label information is less, one might take an active learning approach where the machine learning model can learn with smart human supervision or a completely unsupervised approach where the label information is not present. However, there is a lack of methods in literature which undertakes the problem of change detection partially labelled hyperspectral images. Moreover, if there is a complete absence of labels, the existing methods often fail to grasp minute changes surrounded by large unchanged areas. Hence, the proposed work explores building two systems for detecting changes between a pair of such bi-temporal co-registered hyperspectral images where the label information is completely or partially absent. For this, the work has been explored in an unsupervised and active learning setting. It is found that the proposed methods for both the settings clearly outperformed the state-of-the-art methods in change detection for all the datasets.

The third objective of the thesis involves proving the relevance of AEs for time-series datasets. Stock prices are highly volatile and sudden changes in trends are often very problematic for traditional forecasting models to handle. The standard Long Short Term Memory (LSTM) networks are regarded as the state-of-the-art models for such predictions. But, these models fail to handle sudden and drastic changes in the price trend. Moreover, there are some inherent constraints with the open, high, low and close (OHLC) prices of the stocks. No method in the literature has handled the inherent property of OHLC prices into their consideration yet. We argue that predicting the OHLC prices for the next day is much more informative than predicting the trends of the stocks. This is because the trend is mostly calculated using these OHLC prices only. The problem mainly is focused on Buy-Today Sell-Tomorrow (BTST) trading. In this regard, AEs when pre-trained with the stock prices, may be beneficial. Thus, the contribution of this chapter is a novel framework where a pre-trained encoder is cascaded in front of the multi-task predictor network. This hybrid network can leverage the power of a combination of networks and can both handle the OHLC constraints as well as capture any sudden drastic changes in the prices. It is seen that such a network is much more efficient at predicting stock prices. The experiments have been extended to recommend the most profitable and most overbought stocks on the next day. The model has been tested for multiple Indian companies and it is found that the recommendations from the proposed model have not resulted in a single loss for a test period of 300 days.

The fourth contribution of the thesis deals with enhancing the performance of an AE pre-trained classifier in a synchronised distributed DL setting with the parameter server. Our motivation is that when multiple models in the nodes are trained on different parts of the data, and the aggregated weight update is performed in the main parameter server, it behaves like an ensemble system or a social learning system. Such a combination of multiple classifiers can handle data complexity very well and supersede a single sophisticated classifier. However, it has been observed that in a parallel setting, this fact is falsified if one uses a synchronous weight update mechanism with simple averaging at the parameter server. It has been seen that with an increasing number of worker nodes, the performance degrades drastically. This effect has been studied in the context of extreme imbalanced classification (e.g. outlier detection). In this regard, the AE pre-training is explored with an adaptive cost-sensitive re-training of the individual local classifier and it is found that the proposed method is much more reliable than the other methods.

1.5 Organisation of the thesis

In chapter 2, we have described the working of an autoencoder and its different types. Depending upon the number of layers, the number of hidden nodes, and the types of hidden layers, AEs are broadly classified into multiple types. This chapter gives a preliminary overview of the different types of AEs and their subsequent strengths and weaknesses.

In chapter 3, we have handled a straightforward supervised anomaly detection task. This is a challenging task where a DNN cannot be directly applied for classification. It is obvious to see that most of the datasets do not have an exactly equal number of samples for each class. However, there are some tasks like detection of fraudulent transactions, for which class imbalance is overwhelming and one of the classes has a very low (even less than 10% of the entire data) amount of samples. These tasks often fall under outlier detection. Moreover, there are some scenarios where there may be multiple subsets of the outlier class. In such cases, it should be treated as multiple outlier type detection scenarios. In this chapter, we have proposed a system that can efficiently handle all the aforementioned problems. We have used stacked autoencoders to extract features and then used an ensemble of probabilistic neural networks to do a majority voting and detect the outliers. Such a system is seen to have a better and more reliable performance as compared to the other outlier detection systems in most of the datasets tested. It is seen that the use of autoencoders enhanced the outlier detection performance.

Chapter 4 deals with another imbalanced dataset problem where the labels are not present. Binary change detection is a practical example of having such an unlabelled imbalance problem. There are comparatively more unchanged pixels in the image than the number of changed pixels. Finding changes in bi-temporal co-registered hyperspectral images becomes an even more challenging task due to the large number of spectral bands present in the data. Researchers, therefore, try to handle it by reducing dimensions. The proposed work aims to build a binary change detection system for a pair of such bi-temporal co-registered hyperspectral images using the feature extraction property of deep CAEs. To study the efficacy of CAEs as feature extractors, a CAE pre-trained classifier has been developed to classify images of the ASL hand gestures. It was seen that CAEs are efficient as feature extractors. So, the concept of CAE feature extraction was directly applied to the problem of detecting changes in bi-temporal co-registered hyperspectral images. In this regard, a novel feature extraction system using a modified deep convolutional autoencoder was proposed for dimensionality reduction for detecting changes in hyperspectral image pairs. The proposed methodology is completely unsupervised and hence is much more elegant than other supervised or semi-supervised methods. It is seen that the proposed method to reduce the dimensions is highly effective in detecting the changes and gives better performance than all other methods under comparison. The proposed method has been compared for four hyperspectral image pairs and it is seen that it has achieved the best result compared to the methods it has been compared to. For cases where partial label information is present, an active learning-based approach was also investigated along with the convolutional autoencoder-based feature extraction. It is seen that the proposed approach outperformed the other methods even when partial information was present.

Chapter 5 attempts to handle the problem of forecasting chaotic and rapidly changing events in the domain of time-series datasets particularly, forecasting stock prices. Stock price prediction is very difficult because of the highly nonlinear nature of the stock prices that vary vastly over a given period. It becomes even more problematic when the stock prices undergo sudden changes in a very short time. Moreover, there are some constraints with predicting the OHLC price of the stocks for the next day. In order to provide a meaningful prediction, one needs to handle both the constraints of OHLC as well as capture the sudden changes with as less error as possible. Since the prediction of stock prices is a crucial component in making financial decisions, it is very desirable for the prediction model to be the least erroneous. This chapter attempts to construct an efficient BTST stock prediction model that uses the feature extracting abilities of the encoder and decoder of an AE and the time series capturing abilities of a multi-task predictor network. In this chapter, it has been shown that through such a hybrid combination, the prediction obtained for a testing period of around 300 days gave the lowest error compared to other existing models. A statistical analysis of the empirical results has also been done to show the efficiency of the proposed prediction model. To show the efficacy of the proposed model, experimentation has been extended to build a two-fold stock recommender system that would recommend the top investable stocks. In the proposed recommender system, the predicted prices of the next day are used to gather information about the stocks which are very profitable or are overbought. To recommend the most profitable stocks for the next day, the difference between the opening price and the highest price of the day is considered. William's %R is used to recommend the stocks which will be overbought on the next day. It is seen that the proposed system reliably recommended the most profitable and overbought stocks that matched with the actual results on that day.

In chapter 6 the thesis has been extended to the area of distributed DL research. Federated learning weight update mechanisms have been explored for deep learning. For most distributed DL methods, it is assumed that the worker servers (or local servers) hold independent and identically distributed (i.i.d.) data. However, in practical cases, the conditions of i.i.d. may not be fulfilled. There may also arise global class imbalance situations like that of outlier detection where the local servers receive severely imbalanced data and may not get any samples from the minority class. In that case, the DNNs in the local servers will get completely biased towards the majority class that they receive. This would highly impact the learning at the parameter server (which practically does not see any data). It has been observed that in a parallel setting if one uses the existing federated weight update mechanisms at the parameter server, the performance degrades drastically with the increasing number of worker nodes. This is mainly because, with the increasing number of nodes, there is a high chance that one worker node gets a very small portion of the data, either not enough to train the model without overfitting or having a highly imbalanced class distribution. The chapter, hence, proposes a workaround to this problem by introducing the concept of adaptive cost-sensitive momentum averaging. It is seen that for the proposed system, there was no to minimal degradation in performance while most of the other methods hit their bottom performance before that.

Chapter 7 concludes the work with discussions on some open areas and pitfalls of the proposed methods which were not addressed in the thesis. Some possible future directions of research are also mentioned in this chapter.

Chapter 2

Background

2.1 Autoencoders

One of the main goals of machine learning is to 'learn' or infer from the data automatically without explicitly programming the task. Thus, the quality of the learnt model depends on the type of data it receives. Some tasks are privileged enough to have good quality annotations or labels to the data. But, for many practical tasks, the quantity of labelled data is small or is completely absent. In that scenario, we may want to use the unlabelled dataset to bring out some underlying properties of the attributes or features. Autoencoders (AEs) are a type of approach to learning these underlying properties in an unsupervised fashion. This underlying property may be used for various purposes like classification, clustering, regression, etc. This chapter will explain the concept of AEs and their various forms. The AE is an efficient NN model which is capable of modelling compact and meaningful data representations. Each data sample is transformed by an AE across several layers of non-linearity [21]. This mapping is independent of the actual class information, i.e. the AE requires no labels for the class and is therefore trained using an unsupervised learning process. The AE is forced to reconstruct the input while passing it through some bottle-neck layer (or layers) [Figure 2.1]. If the input is $\mathbf{x}$, then the output $\mathbf{y} = \mathbf{x}$.

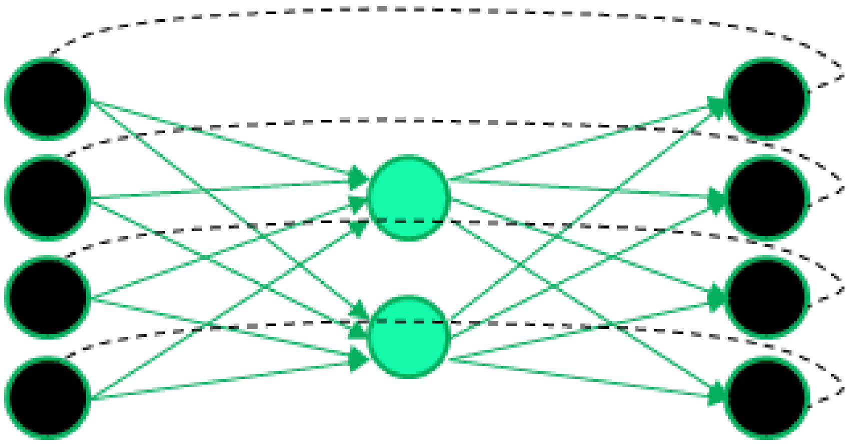


Figure 2.1: Schematic diagram of an autoencoder

The standard method for training is the same as any other MLNN: Backpropagation (BP) [22]. However, other training methods also exist like extreme learning machines (ELMs) [23], evolutionary algorithms (EAs) [24], etc. Training an AE usually includes reducing the error of reconstruction ($||\mathbf{y} - \mathbf{x}||^2$). Since the desired output is the same as the input, the training can be referred to as self-supervised learning.

2.2 Literature Survey

It was LeCun's PhD thesis [25] which presented the original concept of an AE. Some argue that the idea of an autoencoder was proposed much earlier by Cottrell, Munro and Zipser in 1985 [26]. The idea was fairly the same where the data was passed through a bottleneck and was reconstructed at the output. An AE usually consists of two basic elements or parts: an encoder and a decoder (Figure 2.1), which are the usual NN-based connectionist models. The encoder and decoder may be represented as two independent modules which incorporate two functions $f(\mathbf{x})$ and $g(f(\mathbf{x}))$. The task of the function $f(\mathbf{x})$ is to map the data point $\mathbf{x}$ from the original feature space $\mathbb{R}^{\mathrm{N}}$ to some different (latent) feature space $\mathbb{R}^{\mathrm{H}}$, while $g(f(\mathbf{x}))$ would try to generate a reconstruction of $\mathbf{x}$ by mapping $f(\mathbf{x})$ from the latent feature space $\mathbb{R}^{\mathrm{H}}$ to the original feature space $\mathbb{R}^{\mathrm{N}}$. The two functions $f(.)$ and $g(.)$ in contemporary AEs are composed of non-linear functions like sigmoid, relu, tanh [27]. As per the application perspective, it is noteworthy that it is an undesirable trait if the AE is simply used to copy the input. In other words, AEs are often constrained to enable them to learn an approximate version of the input. The problem is formally defined as learning the functions $f : \mathbb{R}^{\mathrm{N}} \rightarrow \mathbb{R}^{\mathrm{H}}$ and $g : \mathbb{R}^{\mathrm{H}} \rightarrow \mathbb{R}^{\mathrm{N}}$ such that they satisfy,

$$arg\ min_{f,g} E[\Delta(\mathbf{x}, g(f(\mathbf{x}))] \tag{2.1}$$

where E is the expectation value of the distribution of $\mathbf{x}$ and Δ is the reconstruction error between the input and the output. Δ is usually chosen to be the mean-squared error, but it may also be varied as per the requirements of the problem. If the functions $f(.)$ and $g(.)$ and linear, then a linear AE is constructed which is said to achieve the same latent feature space as a principal component analysis (PCA) would [28].

The main beauty of an AE lies in its ability to learn latent representations. Real-world data often appear with a sparse high-dimensional representation. If the ML models are trained to operate in such a large feature space, it will frequently lead to a curse of dimensionality [7]. The objective of the reduction of dimensionality is to discover a small, multidimensional space called "intrinsic dimension." The PCA [29] is a conventional technique to dimensional reduction. It involves a linear projection onto a smaller dimension, thus the squared error of reconstruction is reduced. Nonlinear techniques like AEs, however, can and frequently yield better results. It has been shown [30] that auto-associative networks like AEs are equivalent to a non-linear variant of PCA. However, AEs are more generalised. There are other different approaches for learning the latent feature space too [31]. The use of AE to learn the intrinsic dimension is pretty straightforward and its unsupervised nature is one of its beauty. This unsupervised learning of features is designed to identify suitable data representations for classification, reconstruction, visualisation and more. Recently, substantial achievements have been achieved in deep networks due to the feature extracting capabilities of AEs like stacked autoencoders (SAEs) and deep-belief networks (DBNs) that match the present state of the art [10, 32, 33].

Despite the advances, learning robust features is still confronted by issues such as noise outliers that are often seen in real-world data. However, it is seen that AEs can efficiently learn to map outliers differently than the inliers in the data [16]. This is quite an attractive property which has found applications in multiple domains [34, 35, 36]. As AEs try to learn only the important features which may be beneficial for reconstruction, they naturally discard unnecessary variations which come due to noise. The noise in the dataset may be further reduced by an AE using a denoising AE [8, 9]. This AE may develop resilient representations and achieve high performance under various sorts of noises by corrupting input data and trying to reconstruct the clean data at the output.

Feature extraction using AEs has therefore found its way into almost all modern applications like intrusion detection [37], medical image analysis [38], tactile robotics [39], drug discovery [40], etc. It is, however, noteworthy that just like any other ML technique, AE is not an elixir for all problems and might need adaptations to suit the task. Thus, several variations of AEs exist which try to handle different issues from different perspectives.

2.3 Types of Autoencoders

Depending upon the number of layers, the number of hidden nodes, and the types of hidden layers, AEs are broadly classified into multiple types. The following sections describe the different types of AEs and their subsequent strengths and weaknesses.

2.3.1 Types Based on Number of Hidden Layers

The AE architecture may have a single hidden layer or multiple hidden layers. Depending on the number of hidden layers, one may construct a deep autoencoder or a stacked autoencoder.

2.3.1.1 Deep Autoencoder (DAE)

AEs are usually simply taught with a single layer decoder and a layered decoder. This is, however, not a necessity. In reality, it offers numerous benefits to employ deeper architectures of encoder and decoder. The benefits of incorporating depth have been already described in section 1.1 of Chapter 1. Since AEs are basically feed-forward NNs, these benefits apply to AEs as well. The computing cost of some arbitrary functions may be reduced exponentially by increasing depth [41]. Depth can also lower the quantity of training data needed to learn certain functions [15]. A deep autoencoder (DAE) [42] consists of a symmetrical pair of encoder and decoder [Figure 2.2]. The encoder and decoder have more than one layer of weights and the training is usually done on the whole network through BP in the same way as a single-layer NN. It is simply an MLNN whose task is to reconstruct the input back.

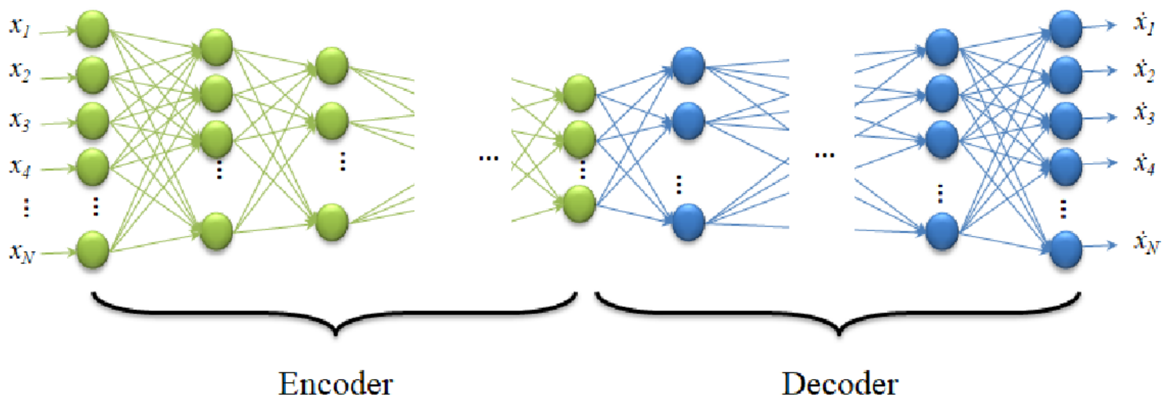


Figure 2.2: Schematic diagram of a deep autoencoder

DAEs are efficient feature extractors and the middle-most layer is used as the code layer whose output is usually taken as the extracted features for a given data. However, if the DAE is too deep (i.e. has a large number of layers), it will again suffer from the problem of vanishing and exploding gradients. In that case, the feature extracting abilities of a DAE will be degraded.

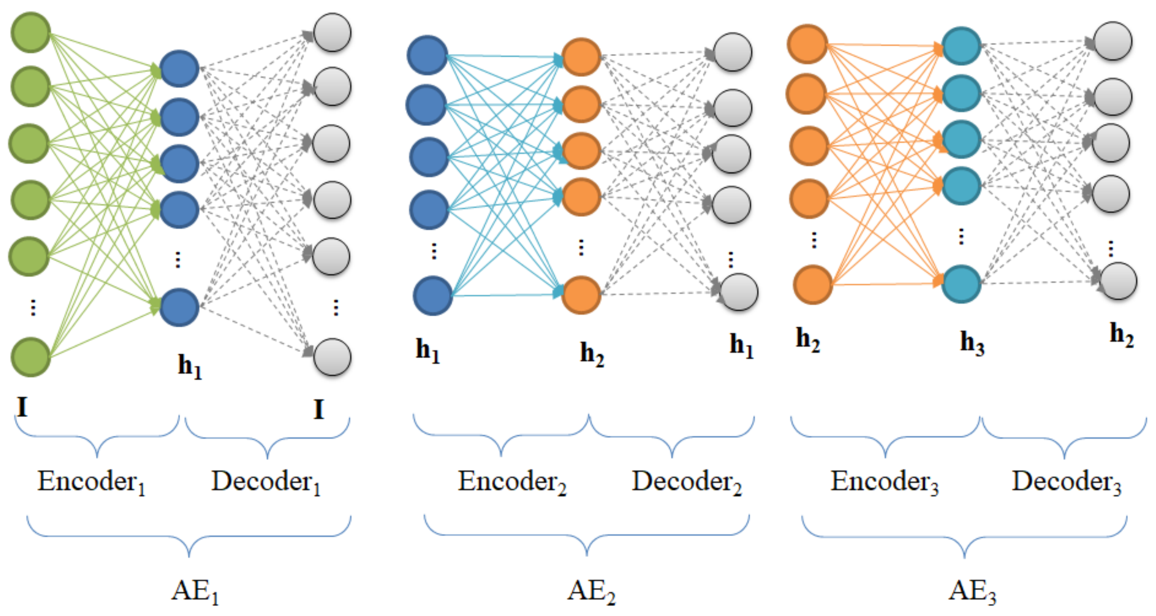


Figure 2.3: Schematic diagram of a greedy layer-wise training of stacked autoencoders

2.3.1.2 Stacked Autoencoder (SAE)

Stacked autoencoders (SAEs) [17] are trained in a greedy, layer-by-layer manner. They are mostly used for pre-training a deep network. In an SAE, to reduce the reconstruction loss of the second layer, the output of the first layer is considered and not the actual input. Suppose we wish to build a DNN classifier which has 5 layers having an input layer with I number of nodes, 3 hidden layers of h_1, h_2 and h_3 nodes respectively and an output layer of o number of nodes corresponding to the number of classes.

Then we would have to train three AEs which would look something like Figure 2.3. The encoders of the three AEs will be stacked successively to get the deep network [Figure 2.4]. The decoders are discarded.

Any DNN which has been constituted by stacking multiple AEs is nothing but a cascade of successive shallow NNs. Unless the DNN is re-trained with the class labels, the classification accuracy will not be good for such DNNs. However, SAEs may be used as good feature extractors as they successively learn the features in a greedy layer-wise manner and can be stacked to form a very deep network [17].

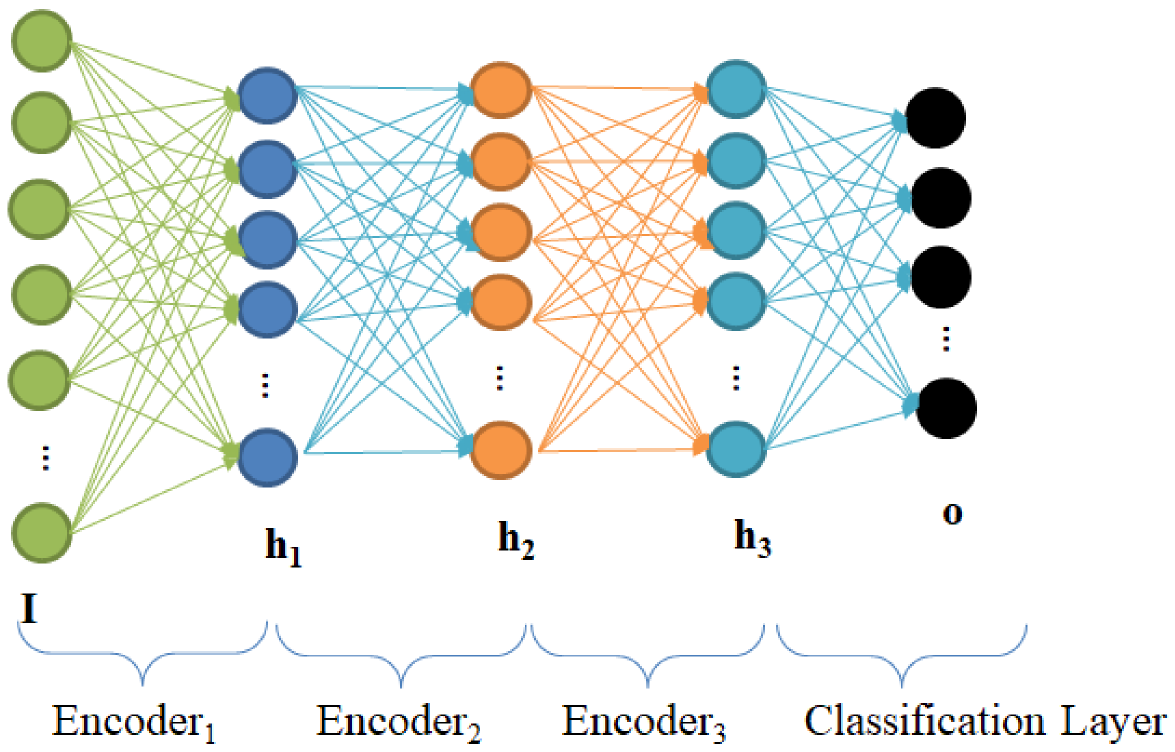


Figure 2.4: Schematic diagram of a deep network composed of stacked autoencoders

2.3.2 Types Based on Number of Hidden Nodes

It is argued that for any NN if the number of nodes in the hidden layer is kept the same or more than the number of nodes in the input layer, then such an NN will learn only identity function and will not be able to properly learn the underlying features [15]. An autoencoder learns to approximate the identity function while passing through a bottleneck. In other words, the features learned in the middle code layer of an AE theoretically should be enough to reconstruct the input back. The identity function appears to be especially easy to be learned; yet we can find intriguing features by imposing restrictions on the network, for example by restricting the number of nodes in the hidden layers. But even when the number of nodes in the hidden layers is high (maybe even higher than the number of input nodes), such intriguing features can still be found by imposing additional network limitations like sparsity. In this context, AEs can be classified as under-complete and over-complete.

2.3.2.1 Under-Complete

When data is passed through such a network, it first compresses (encodes) the input vector to "fit" in a smaller representation and then tries to reconstruct (decode) it back. The task of training is to minimize an error or reconstruction, i.e. find the most efficient compact representation (encoding) for input data, by passing it through a bottleneck [Figure 2.5].

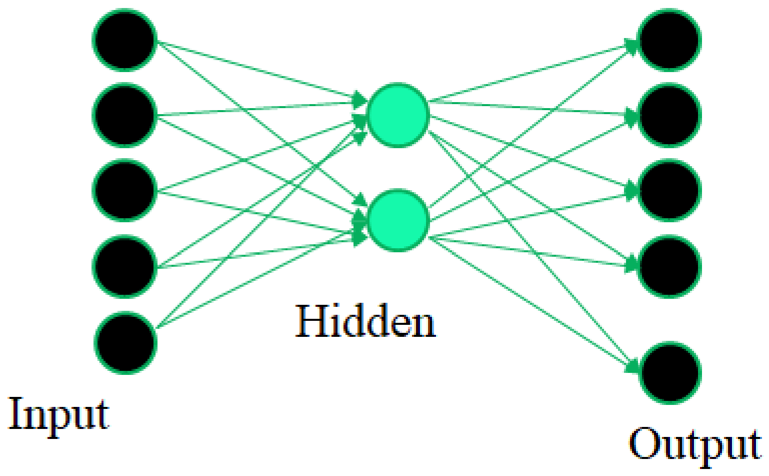


Figure 2.5: Schematic diagram of an under-complete autoencoder

2.3.2.2 Over-Complete

When data is passed through such a network, it first transforms (encodes) the input vector to "expand" in a much larger representation and then tries to reconstruct (decode) it back. We can impose a "sparsity" constraint [43] on the hidden layer, which would prevent it from mimicking the identity function. Such a network is also known as a sparse autoencoder [Figure 2.6]. Normally, a node in a network is said to be active if its output value is one. If the output becomes zero, then such a node will be deemed inactive. Adding a sparsity constraint would basically imply that most of the nodes will remain inactive or dormant and only a few nodes will be active. Such an AE is capable of discovering interesting features in the data because such a network is forced to selectively

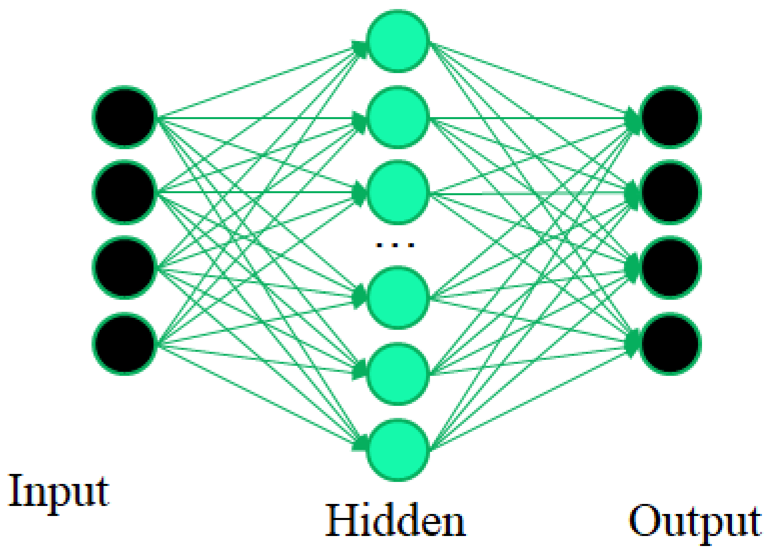


Figure 2.6: Schematic diagram of an over-complete autoencoder

activate only those neurons which contribute to extracting meaningful information from the input data.

2.3.3 Other Types

Some distinct variations of the generic AE architecture exist in order to make sure that the transformed data represents the most significant features of the original input. Depending upon the type of data and problem complexity, many suitable types of AEs have emerged. The following segment describes some of the popular AEs that are used.

2.3.3.1 Fully Connected Autoencoders

Fully connected AEs or vanilla AEs are simple multi-layered networks where each neuron in the former layer is connected to each neuron in the next layer. Examples of such AEs are pictorially depicted in Figures 2.2, 2.5 and 2.6. They are called fully connected because no connection between two neurons in two successive layers is dropped. This kind of network architecture has the maximum number of connections or weights to train among AEs.

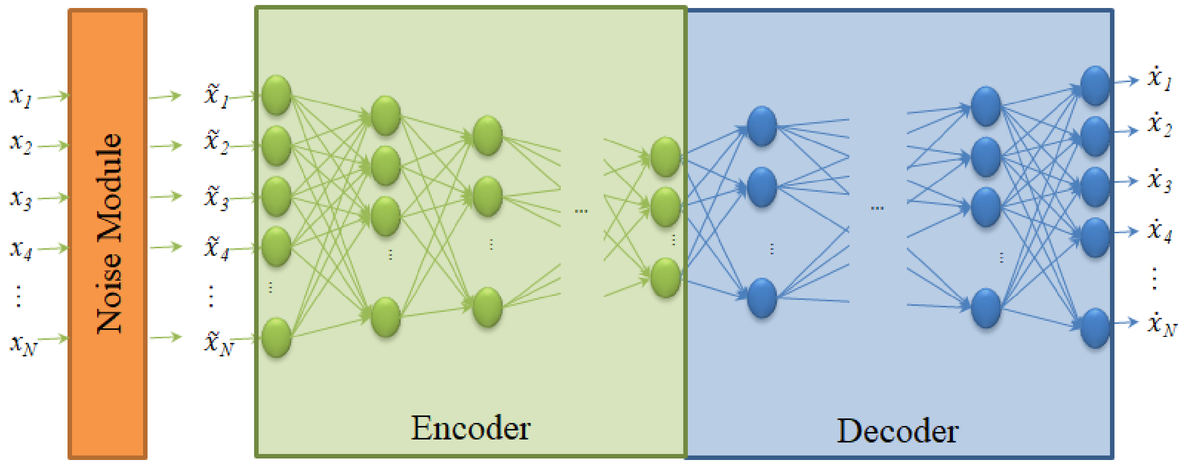


Figure 2.7: Schematic diagram of a denoising autoencoder

2.3.3.2 Denoising Autoencoders

A denoising autoencoder [8, 9] is a special type of AE that addresses the problem of generalisation by randomly adding noise to the input values and corrupting the data intentionally. This noise may be Gaussian, Poisson, salt-pepper, etc. The percentage of corruption introduced is usually about 20-50% [8]. The optimum percentage of corruption depends on the number of samples and the dimensionality. The original input is passed through a noise (or corruption) module which introduces random noise into the features [Figure 2.7]. So, the features $\mathbf{x} = \{x_1, x_2, x_3, ...x_N\}$ for an N-dimensional input now become $\tilde{\mathbf{x}} = \{\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, ...\tilde{x}_N\}$, where $\tilde{x}_i = x_i$ if the pixel is not corrupted and $\tilde{x}_i \neq x_i$ if it is corrupted. The AE then tries to reconstruct back the original input $\mathbf{x}$. Let the reconstructed output for the AE is $\dot{\mathbf{x}}$, then the AE would try to reduce the error of reconstruction given by,

$$E_{reconstruction} = ||\mathbf{x} - \dot{\mathbf{x}}||^2. \tag{2.2}$$

In this case, it becomes crucial to reconstruct the original input by computing the Loss function with the actual values and not the corrupted values. This eliminates the possibility of learning identity function and forces the network to extract latent characteristics. The encoder and decoder may not always need to be fully connected layers but can be

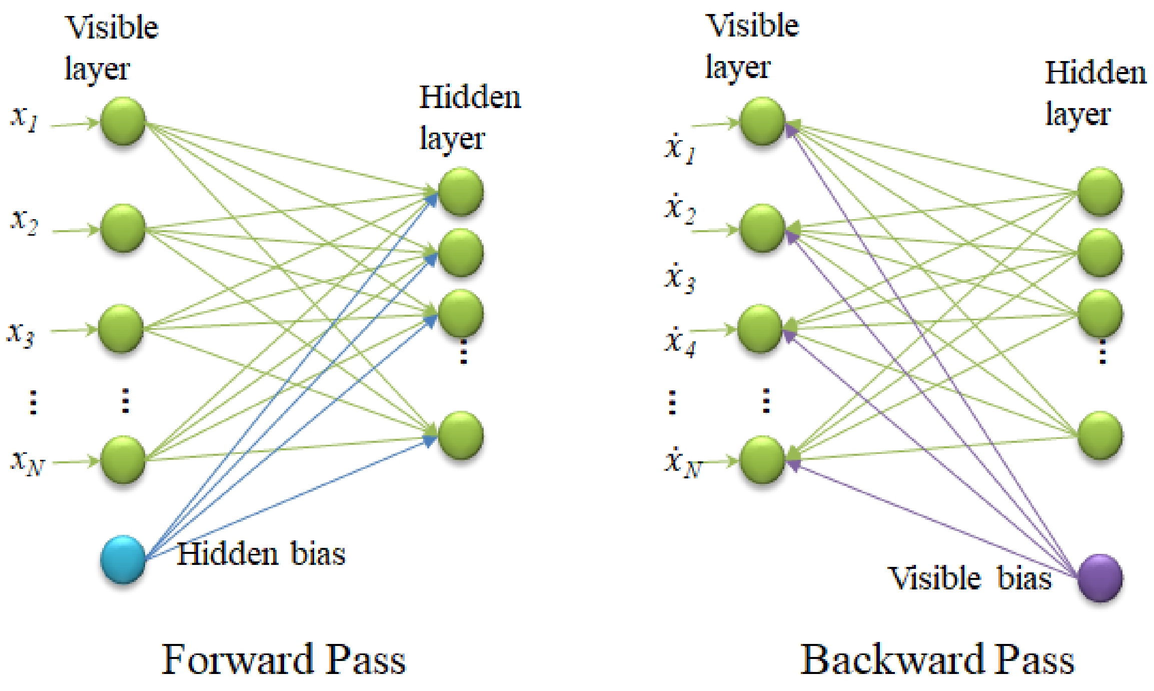


Figure 2.8: Schematic diagram of forward and backward pass in a restricted boltzmann machine

made of convolution layers too.

2.3.3.3 Restricted Boltzmann Machines (RBMs)

Boltzmann Machines (BMs) [44] are stochastic energy-based neural network models which are named after the Boltzmann distribution. Since they are generative in nature, they are able to develop efficient feature representations and tackle challenging combinatorial challenges (given adequate time). BMs are regarded as generative DNNs that include merely two types of layers – visible and hidden. No output nodes are available. A BM contains connections between the nodes of the same layer, contrary to other conventional networks. Restricted Boltzmann Machines (RBMs) [45] are a particular class of BMs and are 'restricted' by the fact that the nodes in the same layer have no connections between them. Like a BM, though, RBMs are also neural networks of two layers and these two layers are connected like a bipartite graph. This indicates that each node of the visible layer is linked to each node in the hidden layer, but there are no connections between nodes in the same layer. Stacking RBMs may constitute a deep belief network (DBN) [10].

There are two more bias weights (hidden bias and visible bias) present in an RBM which is not there in AEs [Figure 2.8]. That is the difference between RBMs and AEs. The hidden bias of RBM results in the forward pass activation and the visible bias allows RBM to reconstruct the input in the backward pass. There is no connection between the visible nodes and hence no method of transferring information among themselves. So, the reconstruction always differs from the real input. It has the same aim of minimising the reconstruction error,

$$E_{reconstruction} = ||\mathbf{x} - \dot{\mathbf{x}}||^2. \tag{2.3}$$

RBMs are trained using contrastive divergence [10]. Some other models have also emerged following the footsteps of an RBM namely, variational autoencoders (VAEs) [46], generative adversarial networks (GANs) [47], etc.

2.3.3.4 Convolutional Autoencoders (CAEs)

Convolution operation [48] is a standard procedure in image processing. Usually a filter is operated across an image and it results in a filtered image [Figure 2.9]. These filtered images are also called feature maps. In order to get a filtered image of the same dimension as the original image, padding needs to be done. Padding would simply add zero pixels around the border. The general convolution operation in the N-dimensional discrete domain is given as,

$$y(i_1, i_2, ..., i_N) = \sum_{u_1=-\infty}^{\infty} \sum_{u_2=-\infty}^{\infty} ... \sum_{u_N=-\infty}^{\infty} f(u_1, u_2, ..., u_N) \\ x(i_1 - u_1, i_2 - u_2, ..., i_N - u_N), \tag{2.4}$$

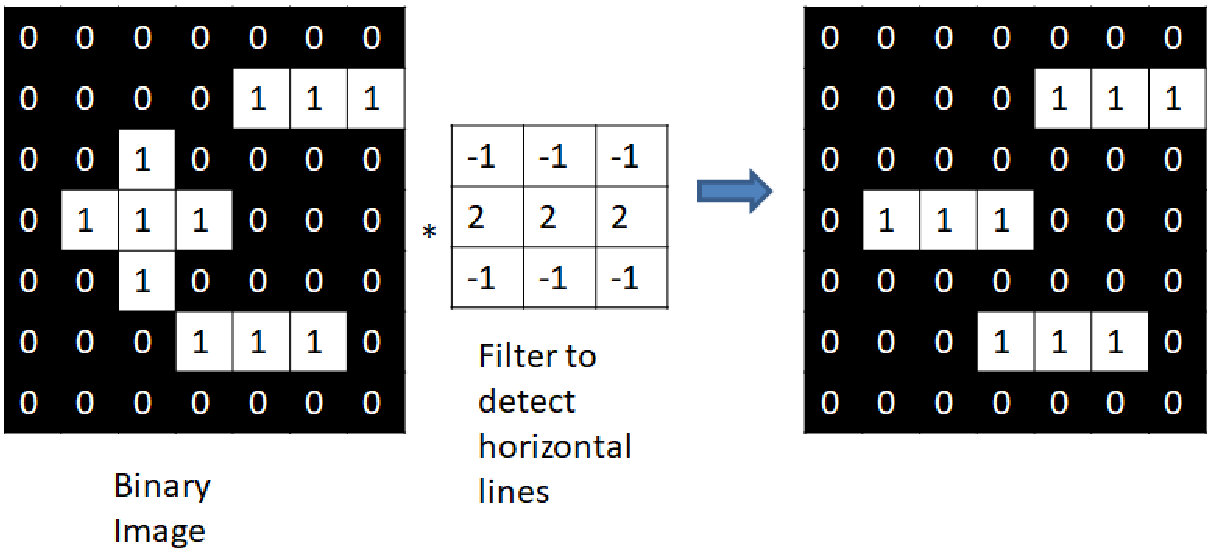


Figure 2.9: A horizontal line detecting filter applied on a binary image (* represents the convolution operator).

where, $f()$ represents the N-dimensional filter, $y()$ represents the N-dimensional output and x represents the N-dimensional input. In the image domain, the signals have finite length. So, for a 2D grayscale image, equation 2.4 becomes,

$$y(i,j) = \sum_{u=-(2k+1)}^{(2k+1)} \sum_{v=-(2k+1)}^{(2k+1)} f(u,v)x(i-u,j-v), \tag{2.5}$$

In order to extract a portion of its content, the convolution operator allows an input signal to be filtered.

In a convolutional neural network (CNN) [49, 50, 51], there are multiple convolution layers. Each convolution layer consists of multiple filters which generate multiple feature maps. A CNN also contains some pooling layers which reduce the size of the images. An $n \times n$ pooling layer will reduce the image dimensions by n. So, if a pooling layer gets an image of dimension $a \times b$, its size will be $\frac{a}{n} \times \frac{b}{n}$ after pooling. Pooling operations are mainly of three types max pooling, min pooling, and average pooling. Figure 2.10 shows the three types of 2×2 pooling operation on an 4×4 image. A CNN consists of multiple such convolutional and pooling layers. If the CNN is trained to reconstruct

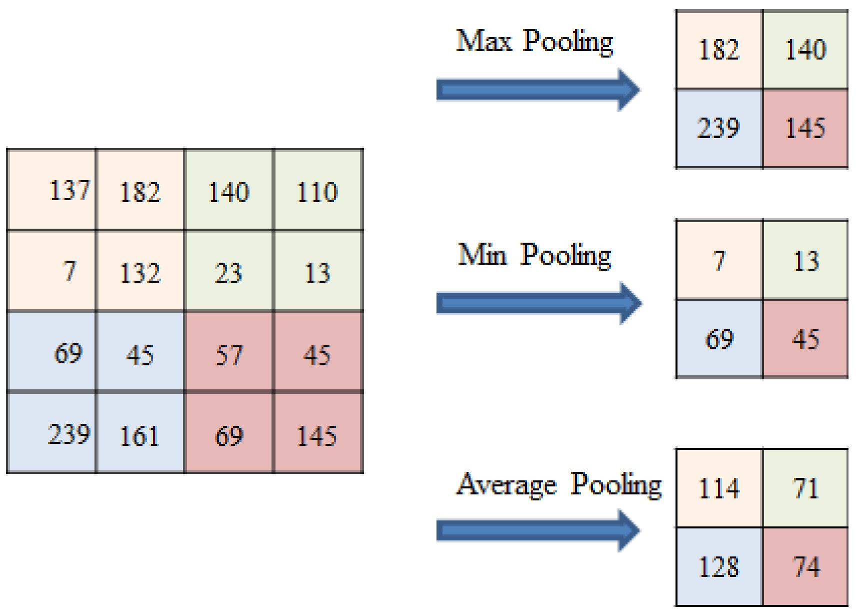


Figure 2.10: An example of the three types of pooling operations.

the input while passing it through several convolutional and pooling layers, then it becomes a Convolutional Autoencoder (CAE) [Figure 2.11]. A CAE has convolutional and pooling layers in the encoder and deconvolutional and upsampling layers in the decoder. Traditional AEs don't take into consideration that a signal may a sum of other signals in their standard formulation. In a fully-connected autoencoder, the input to the network is usually provided in a flattened format where there is no neighbourhood information. For images, the information present in the neighbouring pixels is often relevant in understanding the dynamics of the scene represented by the image. Hence, a convolutional neural network based autoencoder is much suited for tasks concerning image data. Instead of manually calculating the convolution filters $f()$, a CAE learns optimum filters to reduce the reconstruction error. These filters may be applied to any input image to extract features once they have been learned. These characteristics may thus be utilised for all tasks, such as classification or clustering that require a concise representation of the input. CAEs, owing to their convolutionary nature, reach high-dimensional realistic representations because there is always the same amount of parameters needed for creating a feature map, no matter how large the input is. Even though the number of

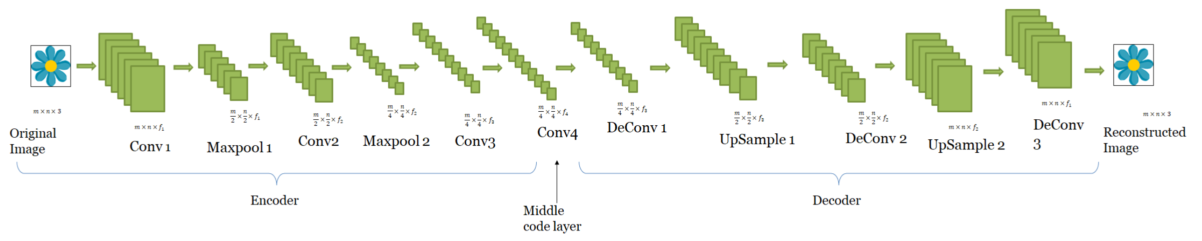

$m \times n \times 3$
Original Image
$m \times n \times f_1$
Conv 1
$\frac{m}{2} \times \frac{n}{2} \times f_1$
Maxpool 1
$\frac{m}{2} \times \frac{n}{2} \times f_2$
Conv2
$\frac{m}{4} \times \frac{n}{4} \times f_2$
Maxpool 2
$\frac{m}{4} \times \frac{n}{4} \times f_3$
Conv3
$\frac{m}{4} \times \frac{n}{4} \times f_4$
Conv4
$\frac{m}{4} \times \frac{n}{4} \times f_3$
DeConv 1
$\frac{m}{2} \times \frac{n}{2} \times f_3$
UpSample 1
$\frac{m}{2} \times \frac{n}{2} \times f_2$
DeConv 2
$m \times n \times f_2$
UpSample 2
$m \times n \times f_1$
DeConv 3
$m \times n \times 3$
Reconstructed Image
Encoder
Middle code layer
Decoder

Figure 2.11: Schematic diagram of a convolutional autoencoder.

parameters in a CAE is less, the number of updates on each filter depends on the size of the image.

2.4 Relevance of Autoencoders in Problems Identified

AE minimises a loss function that reflects the disparity between reconstructed input and the actual input. This optimisation gives an AE the flexibility to extract the latent features for diverse applications. An AE thus automatically detects the underlying structure in an unsupervised manner. The results can subsequently be utilised for extracting important features during classification, clustering, or other prediction tasks.

2.4.1 Outlier Detection

Outlier class modelling is a challenging task in which the majority of the data correspond to the normal data instances and samples that do not correspond to the normal characteristics are identified as outliers or anomalies. This has many practical application areas fraud detections, intrusion detection, detection of novel particles in physics, etc.

This problem is addressed in a wide range of literature. Deep neural networks (DNNs) are being extensively researched to give an answer to the problem of outlier detection [36]. Since a normal multi-layer perceptron is affected by the masking and swamping effects of outliers, it is difficult for any feedforward multi-layer network to handle outliers.

Recently, outlier detection has been achieved using variational AEs [35] and denoising

AEs [52] which demand a good amount of clean and noise-free data for training. The popularity of AEs for outlier detection has gained substantial momentum in recent years [53, 54]. Using AE for this job follows the premise that the latent subspace of the non-outlier (normal) samples is learned by a trained AE. Once taught, an AE would produce a low error of reconstruction for normal samples, whereas for outliers the error will be high. This is a straightforward derivation of the fact proven by researchers [16] that AEs map similar features to similar ranges. Thus, AEs efficiently extract the manifold of the normal class with minimum redundancy and for the outlier class, the mapping range is different from that of the normal class [55]. AEs thus extract those features which differentiate between an outlier and a normal class easily.

2.4.2 Dimensionality Reduction

Autoencoders are easy to employ for dimension reduction [7]. Each under-complete AE in the bottleneck layer efficiently compresses the data to a reduced dimensionality. The original input is projected into the lower dimensional representation by means of the encoder and the decoder uses that lower-dimensional representation to generate a reconstruction of the input. The output of the encoder is the dimensionality reduced feature vector that may be further used for classification or clustering tasks [8].

Different AEs are used for different application areas. AEs can be constructed on the basis of customised architectures of encoders and decoders. They may be shallow with only input and output layers or deep with more layers. The AE may learn more complicated latent coding by increasing architectural complexity. The encoder and decoder can be fully connected or, when the data is in a grid format like images, they can be enhanced using convolution and pooling layers [49].

This property of AEs may be applied to detect meaningful features in hyperspectral images (HSIs) [56]. Binary change detection in bi-temporal co-registered HSIs is a challenging task due to the large number of spectral bands present in the data. Researchers, therefore, try to handle it by reducing dimensions. Moreover for HSIs, manually labelling the pixels becomes challenging due to the sheer number of spectral bands. Since unsupervised methods [57], mainly rely on detecting the points of dissimilarities between the two images, they do not require any manual labelling [58]. Due to this reason, unsupervised methods are mostly less accurate than the supervised [51, 56, 59] or semi-supervised [60, 61, 62] methods but are much more sophisticated in the application. Keeping the problem statement in mind, a solution may be proposed which presents a clear change-map showing all the changes between the two co-registered bi-temporal hyperspectral images. For the purpose of dimensionality reduction, a modified convolutional autoencoder (CAE) may be used as it will preserve both the spatial as well as spectral information in its hidden code layer.

2.4.3 Time-Series Prediction

Multi-variate time-series data are often very difficult to model. This is because of the underlying intrinsic relationships between the predictor variables. AEs in this case applied with a simple assumption that variables that have behaved similarly in the past would behave similarly in the future as well. However, these variables are not always directly linked but have some intrinsic driving factors which may cause fluctuations across time. AEs can efficiently supplement the task of time-series prediction by extracting these intrinsic latent factors [12, 63]. One such domain is the prediction of financial stock prices. In this case, the AEs are trained to reconstruct continuous data. Traditional AE-based stock market prediction models [12] pre-initialise the predictor network (RNN or its variants). However, the basic AE may also be used to supplement the prediction capabilities of the predictor network by efficient learning of latent space.

AE-based latent space representation has not been well studied in the literature on open-high-low-close (OHLC) stock price prediction. This is partly due to the fact that the literature mainly revolves around predicting the direction of the trend using some technical indicators which are basically constructed using these raw OHLC values only. There has been no attempt in the literature as per our knowledge may predict the OHLC prices for the next day directly. Even if a few methods claim to handle this problem, they miss the basic properties of the OHLC values and often end up making meaningless predictions where the predicted low price becomes higher than the predicted high price or the predicted opening and closing prices may not fall in between the predicted low and the high prices. Fluctuating stock prices with these constraints are very difficult to model and often fall prey to bad choices of normalisation. Moreover, when the stock prices suddenly rise or fall drastically in value, most of the models fail to capture these sudden events. Since AEs capture the relationship between variables in their latent feature space, it is wise to explore AEs for these problems.

2.4.4 Pre-initialising Deep Neural Networks

DNNs are more likely to overfit which may be identified if the same architecture is applied to unknown data (or to a cross-validation test). Research is therefore focused on the exploration of unsupervised pre-training approaches based on a latent representation of the lower dimensions generated by AE [13]. This method is based on the idea that redundant information is reduced in lower-dimensional features extracted by an AE and the possible noise occurring due to collection and processing mistakes in the data set may be removed, thereby reducing the chance of overfitting [33, 64]. In this regard, AE pre-trained distributed DL architecture needs to be explored.

In addition, pre-training using AEs and the use of learned weights as pre-initialisation of the supervised classifier network often increase the model's generalisation capabilities and performance. This becomes of particular importance when there are relatively few labelled samples available. In that case, plenty of unlabeled data samples may be used to learn the features and pre-initialise the network to a sub-optimal range. Re-training using the labelled data in such cases is seen as fine-tuning the network according to the class information. Researchers have shown the accuracy of classification can routinely be increased by a substantial margin by using AE weights for pre-initialisation [65].

2.5 Autoencoders versus other Unsupervised Feature Extraction Techniques

The techniques for feature extraction accomplish an alteration of the original feature space to generate a transformed space that is deemed more relevant to the task [31]. Feature extraction can be defined as the task of construction of linear or non-linear combinations of the input feature space with the aim of achieving high discriminating strength between classes.

Principal Component Analysis (PCA) [29, 63] has been one of the mainstream techniques for dimensionality reduction in the ML community. It has been paired with reducing the dimensionality of the data with a basic intuition that the direction having the maximum variance of the data is the one that has the highest separability strength. PCA tries to find such orthogonal axes in the dataset in decreasing order of variance. Thus, the direction with the highest variance is regarded as the first principal axis, the direction with the second-highest variance is regarded as the second principal axis, and so on. Usually for a linearly separable problem, an N-dimensional dataset can be reduced to lesser dimensions by projecting it on n-principal axes ($n < N$). However, for non-linearly separable problems, PCA cannot be implemented straightforwardly. Some initial variants of PCA were proposed like principal curves [66] (where instead of axes, the transformation was based on non-linear curves). It was modified to a sophisticated kernel version called kernel-PCA [67] where a kernel function is used to project the input feature space to a higher dimensional space and the principal components are obtained on that space. PCA is of good interest to the present study because AEs are said to perform a kind of kernel-PCA (k-PCA).

Apart from the standard PCA, there are other non-linear dimensionality reduction methods that have gained attention in ML research. t-SNE (t-Distributed Stochastic Neighbor Embedding) [68] is concerned about retaining the variance of the neighbourhood points in the dataset. This means points which are nearer or far in the original feature space will be nearer or far in the transformed feature space, respectively as well. However, since it is a stochastic method, the results may vary each time the process is run. This is similar to the stochastic nature observed in AE-based transformations.

Neighbourhood-based methods also include Locally Linear Embedding [69] (LLE). It involves finding the k-nearest data points of a sample and then approximating the feature vector of the sample as a weighted linear combination of those k-nearest data points. The weights that are best able to reconstruct the original feature vector are chosen as the optimum. This is analogous to the feature extraction process in AEs where we try to minimise the error of reconstructing the input at the output while passing the data through a bottleneck layer.

Isomap [70] is a spectral-based non-linear dimensionality reduction approach that attempts to retain geodesic distances in the lower dimension. It begins by constructing a neighbourhood network and then calculates the estimated geodesic distance between all pairs of locations. The low dimensional embedding of the dataset is subsequently discovered using eigenvalue decomposition of the geodesic distance matrix.

There is another dimensionality reduction method called random projections (RP) [71] which works by projecting the original features onto a set of a randomly chosen set of axes. This is primarily based on the Johnson-Lindenstrauss lemma which states that for sufficiently high dimensional data there exists a lower dimensional space that approximately preserves the distances between the samples. The trick here is that a simple random matrix can be used to project the features. In Gaussian random projection (GRP) [72], the random matrix is generated using a Gaussian distribution. Unlike PCA, which tries to project the features to a direction of maximum variance and is hence computationally expensive, GRP is very fast and works very well for very high-dimensional datasets.

With all the methods described above, the problem is that there is no guarantee of the feature transformation to be suitable for a particular dataset. With AE in the initial layers, an NN-based model can be retrained with labels along with the final classification layers. Thus, the features extracted by an AE in the pre-training phase can be fine-tuned to the specific task at hand. Although, for unsupervised tasks or classification models which do not learn through backpropagation, an AE-based feature extraction will also be unable to guarantee its suitability. However, for imbalanced datasets, an AE is well suited even though the class labels are not present. This is shown throughout the different chapters of the thesis via various comparisons with the aforementioned dimensionality reduction methods as well as other state-of-the-art techniques which are deemed more suitable for that particular problem at hand.

Chapter 3

Stacked Autoencoders for Supervised Anomaly Detection

3.1 Introduction

Datasets used for PR tasks have several challenges associated with them. The approach to handle such challenges are not always straightforward and need dedicated algorithms. There may arise a case when the datasets are severely imbalanced such that some of the class (or classes) have very few samples and other class (or classes) have comparatively more samples. Such datasets have their own set of issues which make it difficult to detect the rare classes efficiently. These rare occurrences are interesting samples which do not conform to the normal behaviour of the other majority samples. If a model learns the normal behaviour very well, it will ignore the abnormality in most cases. Moreover, since the data is not clearly separable, it often needs some sort of transformation to make the classes separable.

For many complex problems, a linear transformation fails to capture the underlying manifold for each class in the data. So, we resort to non-linear dimensionality reduction techniques. Researchers in [16] have shown that it is a property of AEs to map repetitive structures to a similar range. In this way, AEs can efficiently capture the class-specific data manifold while eliminating the redundancy. For problems like anomaly detection, the redundancies are inevitable as the non-outlier data forms the majority in the data and may form repetitive structures in the underlying manifold which can often make it difficult to differentiate the outliers. For reference, Figure 3.1 can give some idea as to

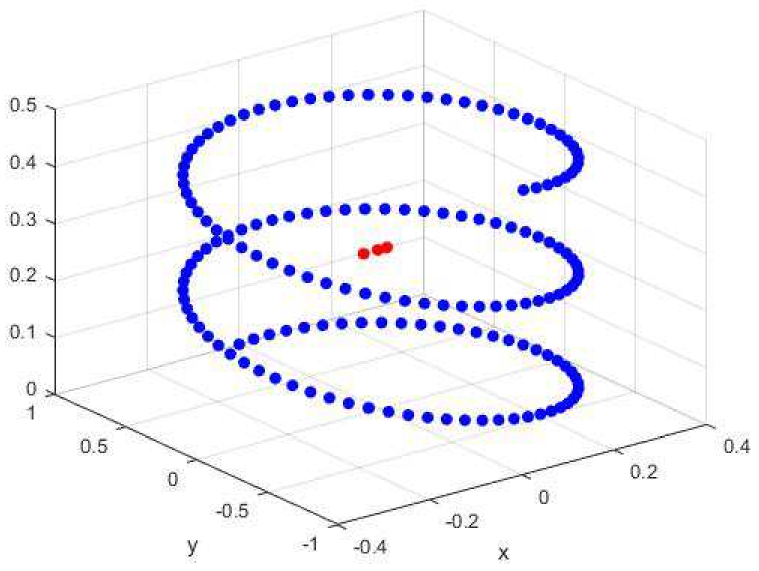


Figure 3.1: An example where the repetitive structures in the underlying manifold of the non-outlier class can wash out outlier effects.

how the repetitive structures in the underlying manifold can wash out outlier effects. The red points are inside the region of the helical structure. But, the distribution followed by the non-outlier class is different from the outliers. In such cases, AEs may find the repetitive structures in the non-outlier class and can map it in almost the same range and outliers to a different range. AEs can enhance the outlier detection mechanism in such cases.

3.2 Outliers

Outliers or anomalies are certain interesting points in data that do not conform to the expected or the natural behavior of the dataset. They represent an observation in the data that is highly unlikely provided a model is built that generates the data [73]. In most of the practical cases, the model is abstract like that of finding a fraudulent credit card transaction in millions of genuine transactions. The data may also have multiple types of outliers like different types of intrusions in a network. One might consider the intrusions as a single outlier class and approach the problem in a binary or a multi-class fashion. However, the practicality of such an approach is questionable as intrusions are highly diverse and may have different reasons to appear in the data. There are many such cases which make single type outlier detection and multiple type outlier detection a crucial part of data analysis process. Figures 3.2 and 3.3 show the two approaches. In the former one, the diversity of the outlier types are not considered and in the latter, the diversity of the outlier types are considered.

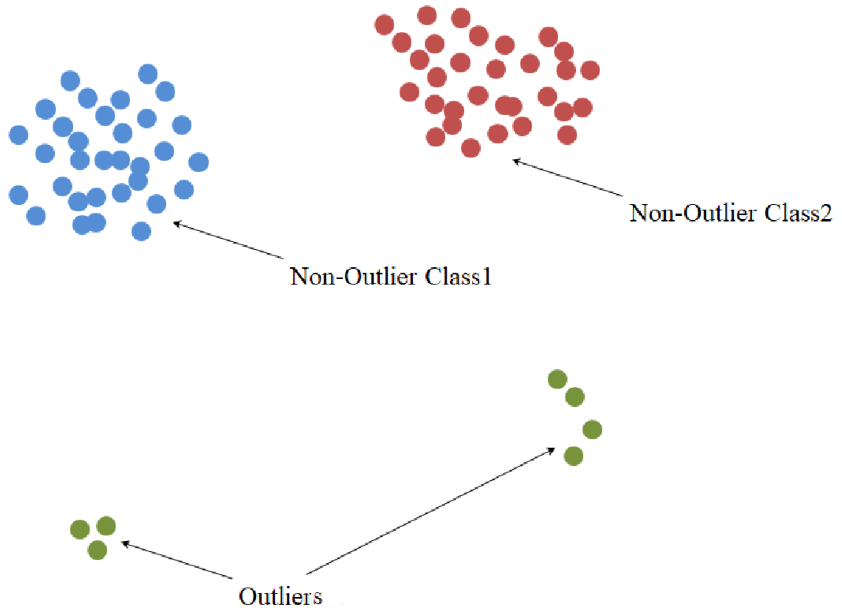


Figure 3.2: Multi-class single outlier scenario

There is however a bottleneck that most of the algorithms get stuck at. Chances of

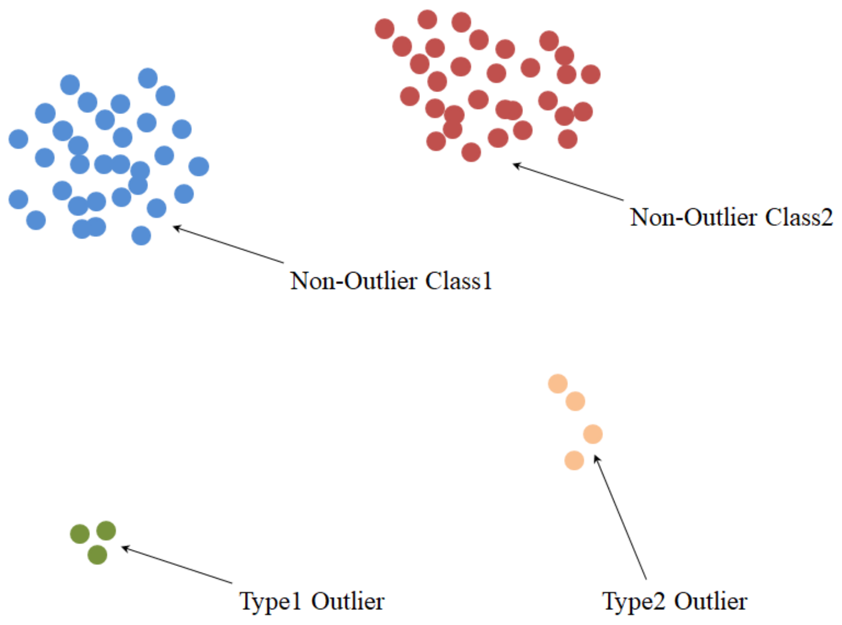


Figure 3.3: Multi-class multiple outlier scenario

finding an outlier in any dataset is extremely rare. In most of the cases, the number of samples from the outlier class is even below 10% of the total number of samples in the entire training set. Identifying them becomes one of the difficult problems in data analysis [74]. Sampling techniques are usually not preferred for outlier detection cases. Oversampling a minority (or an outlier) class or undersampling a majority (or the inlier) class usually affects the generalisation capability [75]. Moreover, the extreme imbalance (below 10%) is a major challenge for many algorithms. This is because the presence of an outlier is often misleading to algorithms like clustering, classification or regression. There may also be cases where the outliers may arise due to different reasons and may have diversity among them. When there are multiple types of outliers in the dataset, each having a different property, taking all the outlier classes as a single class may not make any sense. This differs from a multi-class imbalance problem as in this case the outliers consist of less than 10% of the entire dataset.

Outlier classes and under-represented classes need to be clearly differentiated in that case. Those classes which have rare occurrences in the data (say 10% or less) and do not comply with the normal observations are to be treated as outliers; whereas the classes which are just under-represented (more than 10% but lesser in comparison to the majority class) should be treated as a normal class in a multi-class dataset. Also, we need to consider the fact that there may be more than one outlier classes originating out of different causes. So, there will be three types of datasets handled by the proposed system:

(i) Datasets which have only two classes and one of them is the outlier class.

(ii) Datasets which have multiple classes and one of them is the outlier class.

(iii) Datasets which have multiple classes and some (more than one) of them are the outlier classes.

The classical Random Forests is capable of handling such a situation. This is due to the decision tree technique, it is based upon, which isolates the anomalous observations into small leaves. However, the method again suffers from the presence of redundant features and often cannot capture the underlying manifold for the inlier class. So, we resort to non-linear dimensionality reduction techniques to extract more useful features. Deep neural networks (DNNs), being one of the state of the art techniques, are still being researched to give an answer to the problem of outlier detection. Since, a normal multi-layer perceptron is affected by the masking and swamping effects of outliers, it is difficult for any feedfoward multi-layer network to handle outliers. However, these methods can only be used for datasets with a substantially larger number of samples. DNNs being an extension of the classical neural networks, also suffer from similar masking and swamping effects of outliers. In contrast, models like radial basis functions (RBF) neural networks or probabilistic neural networks (PNN) are structurally more suitable for outlier detection.

3.3 Contribution

This chapter proposes and investigates a new outlier detection framework inspired by the projection methodology through deep learning. It is shown analytically that the proposed method alleviates some of the drawbacks of the several existing standard approaches. We have done multiple experiments on several datasets to prove whether the non-linear transformation given by the AEs or the probabilistic networks used make the method better.

The contributions are as follows:

- A novel outlier detection framework using the projection methodology of stacked autoencoders and probabilistic neural networks.
- An efficient technique to identify outliers in both single type outlier as well as multiple type outlier dataset.
- This chapter also highlights how the use of deep stacked autoencoders can enhance the performance of standard outlier detection techniques.

3.4 Related works

The existing outlier detection methods [76] make different assumptions and hence differ in the way they detect the outliers. Researchers have proposed an abundance of outlier detection techniques [77, 78]. Standard outlier detection techniques mostly include distance-based methods [79, 80], and density-based methods [81]. The problem of outlier detection is usually handled in three types of learning scenarios:

(i) Unsupervised: Learning only from the inlier class (no information of the outlier class is provided),

(ii) Supervised: Learning from the inlier as well as outlier classes, and

(iii) Semi-supervised: Learning only from the inlier class and some unlabeled data (which may or may not be outliers).

Unsupervised methods do not use any information about the outlier class. They include one class classifiers [82, 83], density-based methods [79, 80], clustering based methods [84] etc. One class classifiers try to build a boundary around the inlier class and any sample that lies outside the boundary is classified as an outlier. Density based methods find the density at each region in the feature space and when the density is found to be low, the region is decided to belong to the outliers. Clustering-based approaches [84] identify an object as an outlier if it does not belong to any cluster or if the size of the cluster is too small. All the unsupervised methods are advantageous in the sense that they do not need any outlier information and are hence robust. However, it is to be noted that most of these methods usually set at the outliers to a single group. Supervised methods [78] have an edge on unsupervised methods since they use the outlier information to refine their boundaries. However, unsupervised methods may catch any new anomaly that the supervised method may not as it has not been trained upon that particular type of outlier. Semi-supervised methods [36] are more elegant than the former two where they make use of unlabelled data and train only on inlier data. There is however a claim that using a very small fraction of outliers in the training data is always good for the flexibility of boundaries [83]. Hence, in this chapter, we have considered only the supervised outlier detection methodologies for a fair comparison with the proposed method.

The most popular of the supervised outlier detection algorithms is the classic and simple k-nearest neighbor (kNN) [85]. The popular LOF (local outlier factor) method [78, 86] also makes use of kNN search for each point. It assigns an LOF score to each sample in the dataset irrespective of the data skewness. However, most of the state of the art nearest neighbour based outlier detection methods [87, 88] are sensitive to the parameter k. It means that small change in k can lead to higher misclassification rates [89]. So, we have considered the basic kNN only for comparison purposes as the state of the art nearest neighbour based outlier detection methods are basically based on kNN itself.

Support vector machines (SVMs) have also been extended to one-class learning [90].

Support Vector Data Description (SVDD) [83] has been introduced as a generative approach to the standard SVM. The idea is to build a hypersphere that encompasses all the inlier data and the samples that lie outside are considered outliers. It was shown [83], that SVDD performance are almost comparable to Gaussian, Parzen density estimators and Nearest Neighbor methods. However, it is noteworthy that methods like SVDD (which are strictly one class classifiers) assume an inevitable presence of a fraction of outlier samples in the legitimate non-outlier data [83] to make the decision boundary flexible. However, SVMs are not suited for datasets with large number of samples and take a long time to train. Taking linear kernels may reduce the time complexity, but it makes no sense to use a linear kernel for datasets that have high complexity.

The iForests (Isolation Forests) method [91] has also gained reputation in the outlier detection community. This strategy is exceptionally helpful and is unique compared to every single existing technique. It isolates the outliers considering them to be far from the rest of the observations. It builds up a random forest model with a few decision trees using only a small fixed sized sub-samples, irrespective of the size of the dataset. Therefore, when a forest of such small random decision trees collectively give shorter path lengths for any particular sample, it tends to be an outlier. However, the method's inability to detect outliers in datasets with a high percentage of irrelevant features causes it to fail in most of the cases. It was experimentally found in [55] that dimensionality reduction enhances the classification performance of such methods to a much significant level.

Deep neural networks (DNNs) are being extensively researched to give an answer to the problem of outlier detection [36]. Since, a normal multi-layer perceptron is affected by the masking and swamping effects of outliers, it is difficult for any feedfoward multi-layer network to handle outliers. Researchers have been using self organising maps to handle such problems in unsupervised fashion [92]. Recently, outlier detection have been achieved using variational AEs [35]. However, these methods can only be used for datasets with a substantially larger number of samples. Also, DNNs being just an extension of the classical neural networks also suffer from similar masking and swamping effects of outliers. In contrast, networks like radial basis functions (RBF) neural networks or probabilistic neural networks (PNN) are structurally more suitable for outlier detection [93].

There are many methods which can detect outliers [94] in a multi-class dataset. These methods assume that all the outliers should always belong to a single class, though practical data may have different types of outliers arising out of different causes. Such methods can therefore be regarded as "multi-class single outlier type detection techniques". We have introduced the problem in a different way where the outlier class too can have multiple sub-types along with the multiple types of non-outlier classes. Outlier classes and under-represented classes need to be clearly differentiated in that case. Those classes which have rare occurrences in the data (10% or less) and do not comply with the normal observations are to be treated as outliers; whereas the classes which are just under-represented (more than 10% but lesser in comparison to the majority class) should be treated as a normal class in a multi-class dataset. Also, we need to consider the fact that there may be more than one outlier classes originating out of different abnormal causes. The classical Random Forests [95] is capable of handling such a situation. It is due to the decision tree technique, it is based upon, which isolates the anomalous observations into small leaves.

The proposed method uses a non-linear transformation by stacked AEs [96] and then uses an ensemble of probabilistic neural network to model the outlier class. In doing so, it can efficiently identify outliers in both single type outlier dataset as well as multiple type outlier dataset with better performance than the standard methods used. The proposed method, hence, falls into a category of multi-class multiple outlier type detection technique. There is a lack of dedicated methods in the literature that can handle the outlier detection problem in both multi-class single outlier type detection and multi-class multiple outlier type detection.

3.5 Proposed Method

We have proposed a novel outlier detection technique using deep stacked AEs and probabilistic neural networks. Such a system is shown to provide better results than the state of the art techniques. The proposed method efficiently handles both the single type outlier cases as well as multiple type outlier cases. This section discusses the concepts and strategies employed in the proposed method. In sections 3.5.1 and 3.5.2 we briefly explain the working of a probabilistic neural network and the AE networks. In section 3.5.3 we introduce the proposed method and the various strategies involved to perform efficient outlier detection.

3.5.1 Probabilistic Neural Networks

A Probabilistic Neural Network (PNN) is a representation of a statistical algorithm called Kernel Discriminant Analysis (KDA). The operations are organized into a multilayered feedforward network with mainly three layers namely the input, pattern and summation layers [Figure 3.4]. (In some cases, there is also an output layer, which we have omitted in our model, as we are interested in the immediate class scores rather than hard class labels.) In a way it estimates the probability density function (*pdf*) of classes by using the samples of the training set. In the pattern layer, each node calculates the *pdf* for *each* training sample according to equation 3.1.

$$g(\mathbf{x}) = \frac{1}{\sigma}\mathcal{F}(\frac{\mathbf{x}-\mathbf{x}_i}{\sigma}) \tag{3.1}$$

where $\mathbf{x}$ is the unknown (input), $\mathbf{x}_i = i^{th}$ sample, $\mathcal{F}()$ is the weighting function and σ is the smoothing parameter. In the pattern layer, the nodes (corresponding to the training pattern) are grouped according to the class of the training sample. A single group accounts operation for the next summation node.

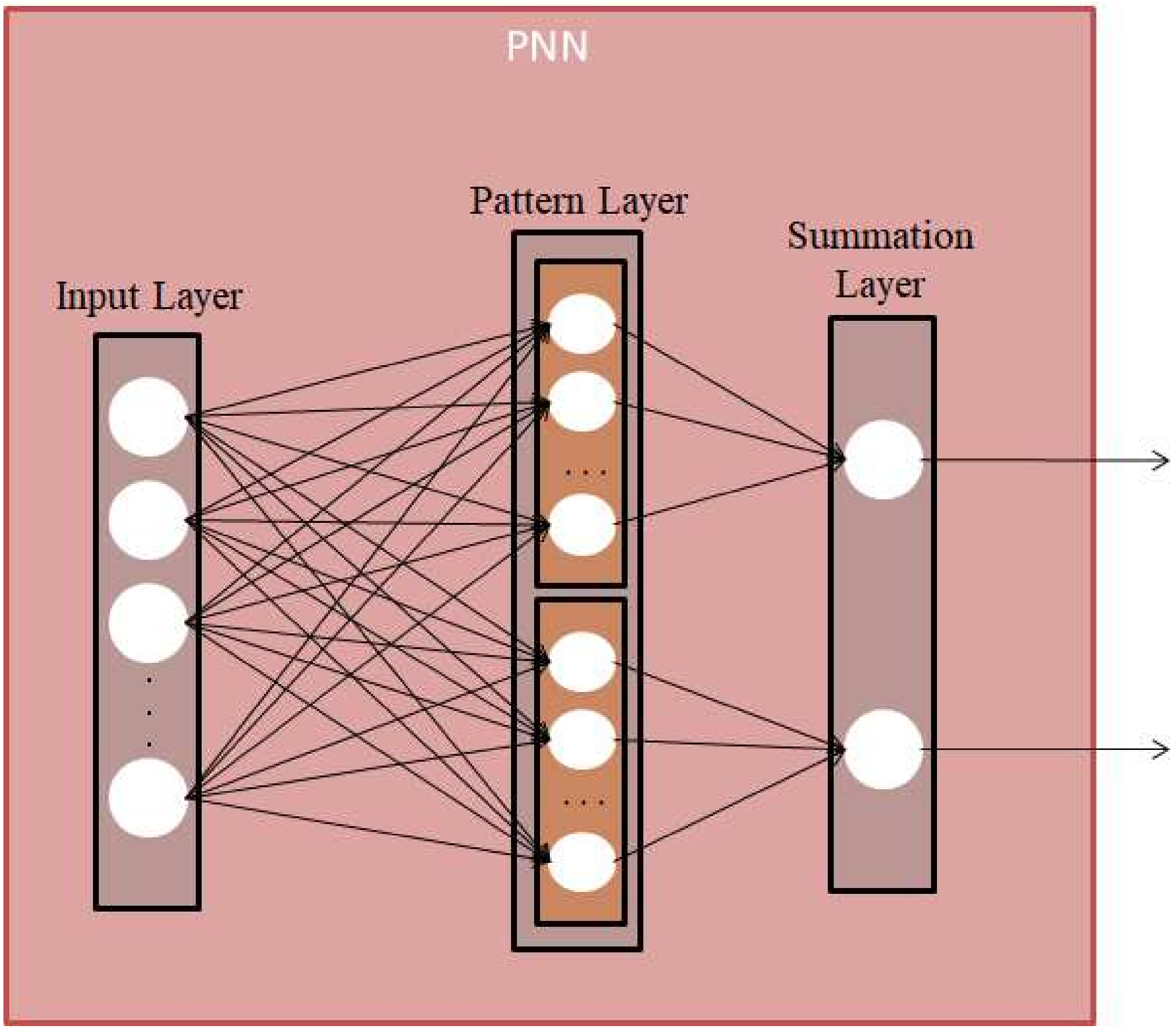


Figure 3.4: A schematic diagram of probabilistic neural network

If there are C classes, the k^{th} summation node sums the values received from the pattern nodes in the k^{th} class, called mixed Gaussians or Parzen windows. The sums are defined by

$$f_k(\mathbf{x}) = \frac{1}{n\sigma}\sum_{i=1}^{n_k}\mathcal{F}(\frac{\mathbf{x}-\mathbf{x}_i}{\sigma}), \tag{3.2}$$

where n_k is the number of samples in the k^{th} class.

The output from the k^{th} node in the summation layer corresponds to the k^{th} class. As newer training patterns arrive, the estimate of $f_k(\mathbf{x})$ is refined without the need of any retraining. As the number of samples increase, each summation node in the PNN tends to get closer to the actual underlying class density functions.

Any input vector $\mathbf{x}$ is put through all the functions $f_k(\mathbf{x})$ and the maximum a posteriori probability (MAP) value of all the $f_k(\mathbf{x})$ decides the class.

Usually, the weighting function $\mathcal{F}()$ is chosen to be a radial basis function (RBF) (also called a kernel function). The radial basis function is so named because the radial distance is the argument to the function. The farther a training sample is from the new unknown point, the less influence it has. A PNN is insensitive to outliers and is hence a very good candidate for outlier detection. PNNs also have an added benefit as new patterns arrive, they are stored into the net. The network improves its generalization and the decision boundary can get more complex. If the number of training patterns become large, the value of σ can be reduced without retraining the network. It was shown by researchers [83] that in case of high sample size, Parzen density estimators are to be preferred over SVMs. Since, each subnetwork of a PNN act as the Parzen density estimator for the particular class, it can easily be used for outlier detection purposes.

However, it was observed that use of PNN alone could not provide better results. Moreover, similar to kNN, such a system has a large storage requirement. We have tried to tackle such a high storage requirement by dimensionality reduction using stacked AEs (SAE).

3.5.2 Autoencoders

An autoencoder (AE) is a type of multi-layer neural network that is trained to reconstruct the input as closely as possible in its output nodes [Figure 3.5].

While stacking the AEs we use the encoder part and the hidden nodes become the input to the next AE network. If the nodes in the hidden layers are lesser in number, they represent a reduced representation of the input and are hence called under-complete

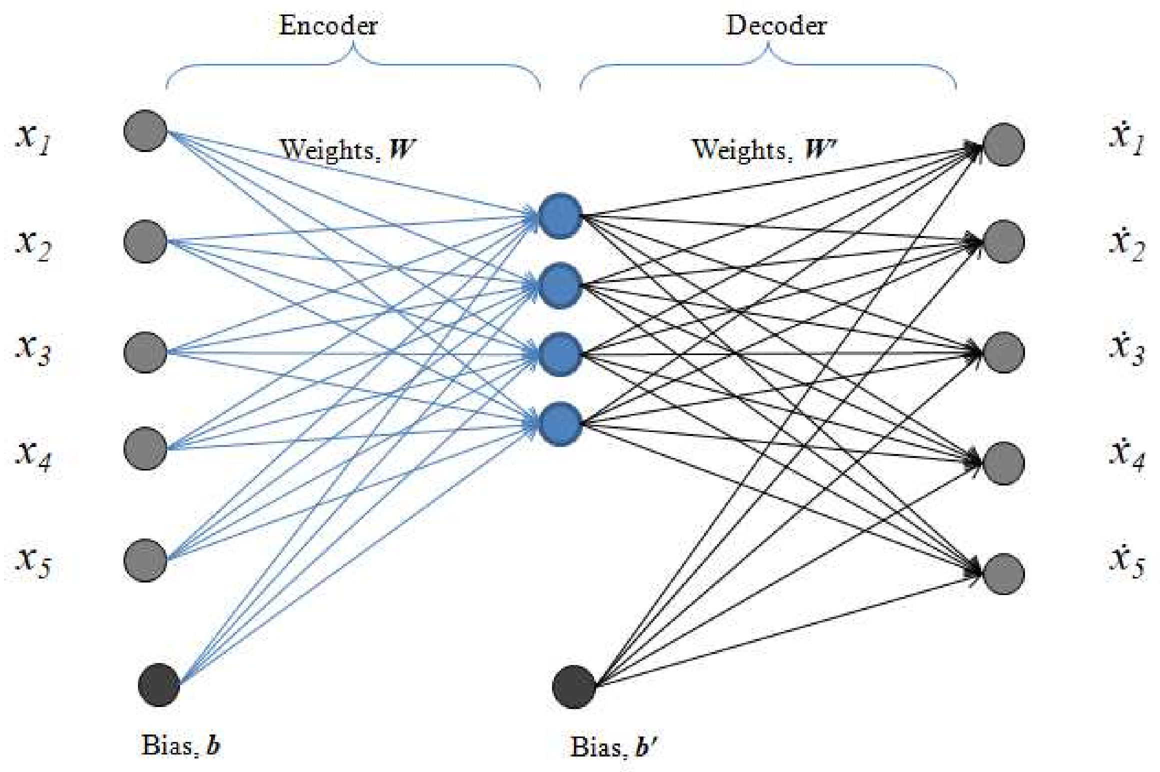


Figure 3.5: A schematic diagram of an under-complete autoencoder

AEs. Such a concept of reduced representation, makes it a natural approach in outliers detection. The basic motivation is that usually the outliers are much harder to be accurately represented in such reduced representation and hence stand out of the normal data distribution.

3.5.3 Stacked Autoencoder Probabilistic Neural Network (SAE-PNN)

Most of the outlier detection methods, employ semi-supervised [77] or unsupervised learning [78, 97]. The semi-supervised models are usually trained using only the inlier class because abnormal or outlier points occur rarely and hence are not easily available. Outliers being only a very small fraction of the data, can never be sufficient enough to construct a binary classifier. However, they can be used to efficiently refine the decision boundary that separates the inlier samples. In the proposed method, we have trained the stacked AEs using both the labelled as well as unlabelled samples. However, for the final training of the ensemble of PNNs, we have used a tiny fraction of the labelled outlier samples along with the majority of the inlier samples. This inclusion of outliers in the training set was done to incorporate a refinement of the model. In case of outlier detection, we need to make a tradeoff between generalization (keeping the inliers inside the decision boundary) and specialization (excluding the outlier patterns outside the boundary). Most of the models used for outlier detection cannot specialize from data unless they are given an internal bias. The proposed model alleviates these pitfalls. However, there were other factors too which needed to be considered.

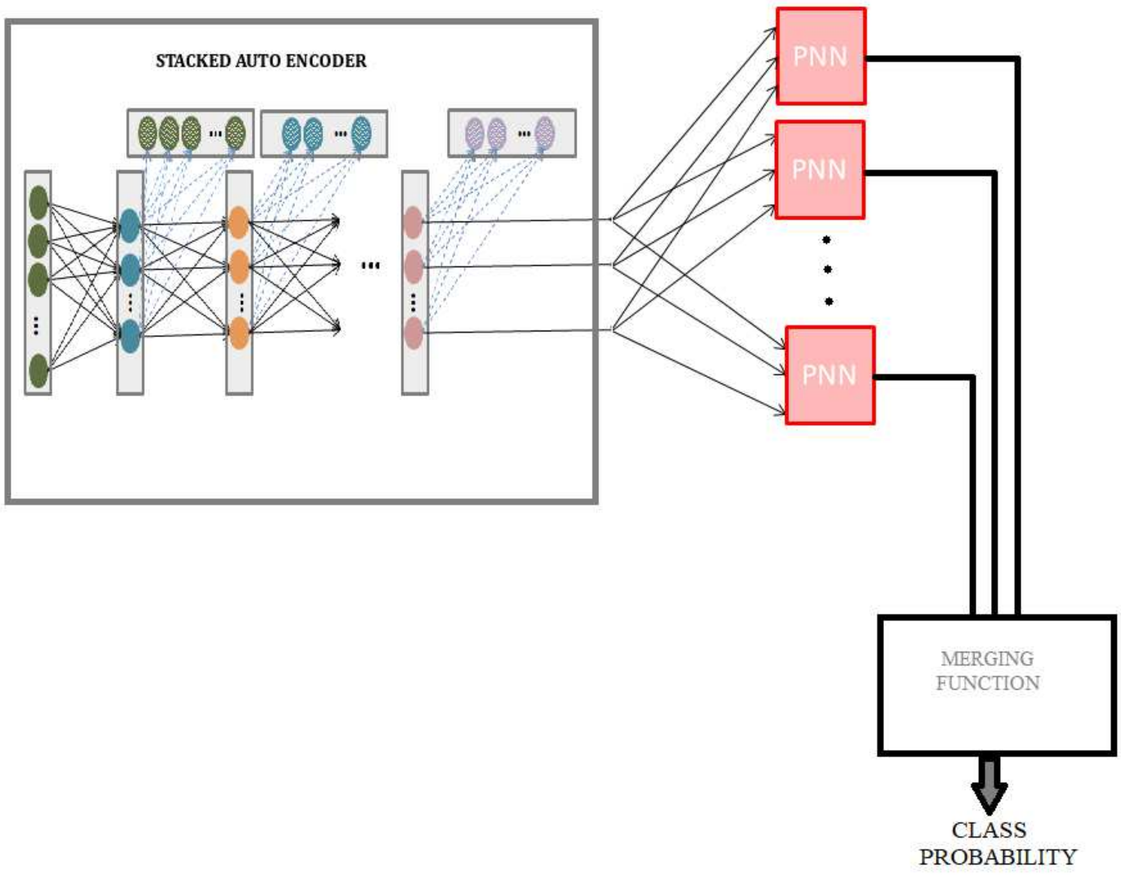


Figure 3.6: A diagrammatic representation of the proposed framework. (The dashed lines suggest that the decoder part of the autoencoder was not used after feature extraction at each stage.)

The peak of the radial basis function in a PNN is always centered at the point it is weighting. The smoothing parameter (σ) determines the spread of the RBF. The primary work of training a PNN is selecting the optimal sigma values to control the spread. Extremely large values of σ cause the model to under-fit and extremely small values cause the model to over-fit. However, it was seen that for imbalanced datasets, lower values of sigma gave lower false alarms and higher values of sigma produced lower misses. Since, both the factors are important in case of efficient anomaly detection, it motivated us to use an ensemble of PNNs, each having a different sigma value, to get the best of the both.

Suppose the outlier set is $\mathcal{O}$ and the inlier set is $\mathcal{I}$ and the original training dataset is $\mathcal{S}$. Then,

$$\mathcal{O} \cup \mathcal{I} = \mathcal{S} \tag{3.3}$$

The set $\mathcal{I}$ is randomly divided into N subsets such that

$$\mathcal{I}_1 \cup \mathcal{I}_2 ... \mathcal{I}_N = \mathcal{I} \tag{3.4}$$

Here, N is the number of PNNs we have used in the ensemble. Each of the i^{th} PNNs is trained with entire set $\mathcal{O}$ and the set $\mathcal{I}_i$. For multi-class multiple outlier type dataset, the majority class training samples are divided into N subsets and the minority classes as well as the outlier classes are taken as a whole.

However, such methods perform better when any dimensionality reduction has been performed on the datasets [55]. We know that AEs provide a non-linear transformation and perform dimensionality reduction efficiently [19]. It was empirically found that stacking more (usually greater than two) number of AEs in the feature extraction stage resulted in better outlier detection rates. Results from section 3.6.3, present this claim. A diagrammatic representation of the proposed framework has been shown in Figure 3.6.

The proposed SAE-PNN model consists of two parts. The first part is a pre-trained stacked autoencoder and the second part is an ensemble of probabilistic networks. The output from the SAE is fed to the ensemble of PNNs. The ensemble technique needs a decision merging function which determines the overall decision jointly made by the models in the ensemble.

The details of the different modules are provided as follows.

3.5.3.1 Stacked Autoencoder

Let us consider after feature extraction using the stacked AE (SAE), each feature vector gets a transformation $SAE(\mathbf{x})$, where,

$$SAE(\mathbf{x}) = \mathcal{E}_1(\mathcal{E}_2(...\mathcal{E}_h(\mathbf{x}))). \tag{3.5}$$

Here $\mathcal{E}_i()$ represents the encoder function at the i^{th} stacked AE and h represent the number of hidden layers used in SAE.

3.5.3.2 Probabilistic Neural Network

The intermediate RBF scores of $\mathbf{x}$ belonging to each of the k classes in the n^{th} PNN was given according to

$$l_k^n((x)) = f_k^n(SAE(x)). \tag{3.6}$$

3.5.3.3 Merging Function

If there were N PNNs used in the ensemble, the final RBF score $\mathbf{x}$ belonging to each of the k classes in the entire ensemble could be obtained according to

$$l_k(\mathbf{x}) = \sum_{n=1}^{N}(l_k^n((x))). \tag{3.7}$$

Now, the classification scores of $\mathbf{x}$ belonging to each of the k classes is calculated as

$$s_k(\mathbf{x}) = \frac{l_k(\mathbf{x}) - min(l_k(\mathbf{x}))}{max(l_k(\mathbf{x})) - min(l_k(\mathbf{x}))}. \tag{3.8}$$

3.5.3.4 Class Probability

The probability of $\mathbf{x}$ belonging to the k^{th} class is obtained as,

$$p_k(\mathbf{x}) = softmax(s_k(\mathbf{x})). \tag{3.9}$$

The node that gives the maximum probability is the true class label for the unknown sample $\mathbf{x}$.

3.6 Experiments

This section is committed to illustrate the abilities of the proposed method in various situations of outlier detection. We have done some thorough experiments as to how the results vary with values of σ and the depth of the SAE network used. We have given a comparative analysis of the performance of our proposed method with some of the standard methods used in outlier detection both with SAE and without SAE to clearly study the effects of using SAE in outlier detection performance. Finally, we also show that the proposed method is capable of handling multiple outlier type dataset.

For fair comparison, the value of k in k-NN was chosen using cross-validation. The number of predictors for both iForests and Random Forests were taken at a default value of 100. The SAE was trained using ‘adadelta’ optimiser and mean squared error was used. The model was pre-trained for 200 epochs.

3.6.1 Datasets used

The datasets used in the experimental analysis of the proposed method are all highly skewed in nature. Table 3.1 shows the various single outlier type datasets used and their properties. The main complexity of each of the datasets lies in the fact that the outlier class is less than 10% of the total samples. We have used 10 fold cross validation to analyse the performance on each of these datasets.

Table 3.1: List of single outlier type data sets used for experimental purpose

Dataset	**Number of Features**	**Number of Samples**	**Percentage of Outliers**
PenDigits [98]	16	6,870	2.27%
OptDigits [99]	64	5,216	3%
Forest Cover [100]	10	286,048	0.9%
MNIST [101]	100	7,603	9.2%
HTRU2 [102]	8	17898	9.2%
SatImage-2 [99]	36	5,803	1.2%
Credit card [103]	29	284,807	0.172%
Speech [104]	400	3,686	1.65%

To analyse the performance of the proposed method on multiple outlier type datasets, we have used four highly skewed datasets namely Statlog (Shuttle), Page Blocks, ANN-Thyroid Disease and Wine Quality (White) from the UCI Machine Learning repository. For the Statlog (Shuttle) and ANN-Thyroid Disease datasets, the training and the testing sets were already separated. For the other two datasets, we have used 70% for training and 30% for testing, so that the outlier fraction remains the same in both the training and the test set.

Table 3.2: Number of features and samples in each class for multiple outlier type datasets

	Datasets			
	Statlog (Shuttle)	**Page Blocks**	**ANN Thyroid Disease**	**Wine Quality (White)**
No. of features	9	10	21	12
Class 1	45586 (78.6%)	4913 (89.8%)	93 (2.47%)	20 (0.41%)
Class 2	50 (0.09%)	329 (6%)	191 (5.06%)	163 (3.32%)
Class 3	171 (0.29%)	28 (0.5%)	3488 (92.47%)	1457 (29.74%)
Class 4	8903 (15.35%)	88 (1.6%)		2198 (44.9%)
Class 5	3267 (5.63%)	115 (2.1%)		880 (17.96%)
Class 6	10 (0.017%)			175 (3.57%)
Class 7	13 (0.023%)			5 (0.1%)

Table 3.3: Confusion matrix for the non-outlier class

		Predicted Class	
		Inliers	**Outliers**
Actual Class	**Inliers**	Hit (True Positive)	False Alarms (False negative)
	Outliers	Missed Alarms (False Positive)	Outlier Hit (True Negative)

As can be seen from Table 3.2, many classes constitute less than 10% of the data and are therefore rarely occurring. However, each of these classes have a different reason to appear in the data. For example, in the Page Blocks dataset there are five classes which represent text, horizontal line, graphic, vertical line and picture, respectively. Majority of the samples belong to text, but the other four classes vary in properties and are not similar to each other to be put into a single outlier class. In this case, there are multiple classes which do not conform to the majority (which in this case is ‘text’) of the data .

3.6.2 Dependence on σ

For any outlier detection problem, the confusion matrix for the non-outlier class looks like FNR (false negative rate) suggests the rate as to how often the classifier predicts a sample as an outlier when actually it is not. It is given by

$$FNR = \frac{False\ Negatives}{True\ Positives + False\ Negatives}. \tag{3.10}$$

TNR (true negative rate) suggests the rate as to how often the classifier predicts a sample as an outlier when actually it is an outlier. It can be represented as

$$TNR = \frac{True\ Negatives}{False\ Positives + True\ Negatives}. \tag{3.11}$$

Any good outlier detection method should be able to give high TNR and low FNR. So, we can say, the detection rate (DR), where

$$DR = (1 - FNR) \times TNR \tag{3.12}$$

is high. It is seen (Figure 3.7) that the value of σ has an optimal region (shaded by green) with respect to the DR curve. Within that region of σ, it is seen that for imbalanced datasets, lower values gave lower false alarm rates (FNR) and higher values gave a lower outlier miss rates (FPR) so that the detection rate is quite high. Figure 3.8 shows how the ensemble of PNNs each having a different sigma value helps us get the best of both. In order to choose the sigma value for each PNN, we have first noted down the plateau region of the sigma for a single PNN as shown in Figure 3.7. Now, if there are N PNNs, the plateau region of sigma is divided equally into N equispaced values. Each of these values are then taken as sigma for each of the N PNNs.

3.6.3 Dependence on number of autoencoders stacked

In this section we have shown an empirical investigation of the effect of stacking more (usually greater than two) number of AEs in the feature extraction stage. Table 3.4 gives the network specifications of stacked AEs used for each dataset. To provide a comparison

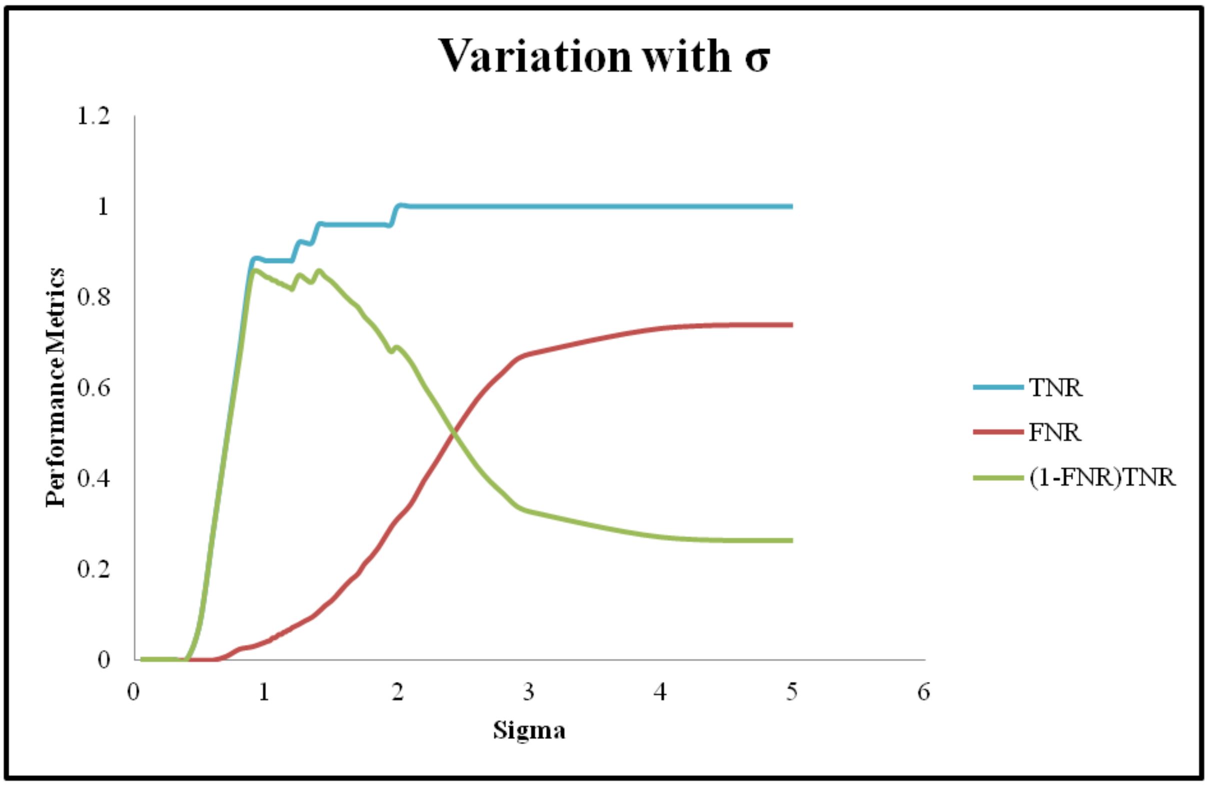


Figure 3.7: Variation of various performance measures (TNR, FNR and $(1 - FNR) \times TNR$) with respect to σ. (Calculations shown are for OptDigits dataset with the autoencoder configuration 64-10-10-10-10-10-10)

regarding the use of AEs with multiple PNNs against using only multiple or single PNNs in the outlier detection problem, we have given a curve that depicts the variation of the ROC-AUC (Area Under the Curve of Receiver Operating Characteristics) with respect to the number of hidden layers. Figure 3.9 shows that the use of more hidden layers in the AEs have significantly enhanced the performance. However, it is also seen that there is an optimal depth beyond which the performance has decreased too. This suggests that use of deep neural networks certainly enhances the performance of the model, but going too deep is not always the best option.

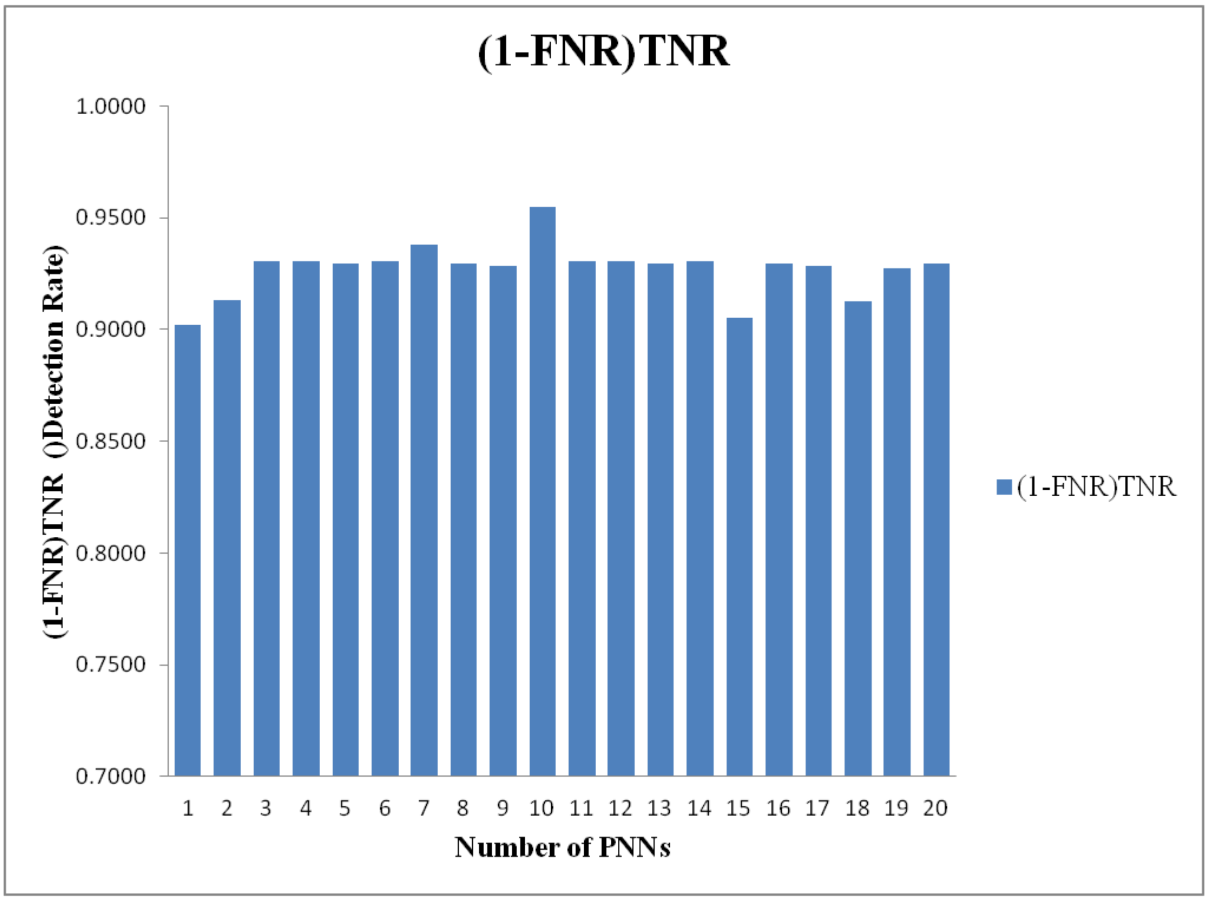


Figure 3.8: Variation of the detection rate ($(1 - FNR) \times TNR$) with respect to number of PNNs used. (Calculations shown are for OptDigits dataset with the autoencoder configuration 64-10-10-10-10-10-10)

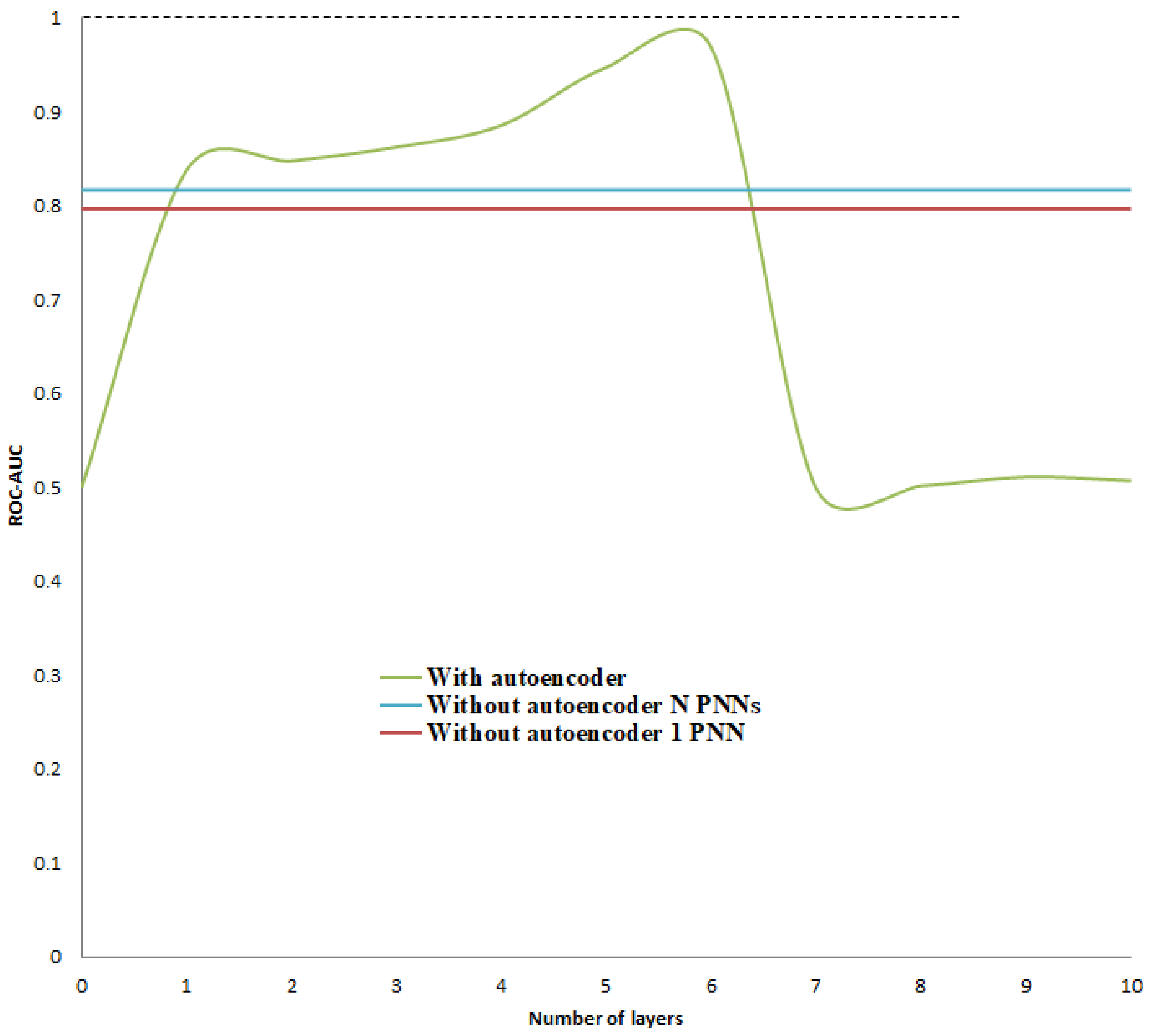


Figure 3.9: Variation of the ROC-AUC with respect to the number of hidden layers used. (Calculations shown are for OptDigits dataset where the number of hidden neurons in each layer is 10)

Table 3.4: Autoencoder configuration used for each dataset

Dataset	Number of Hidden Layers	Nodes in each hidden layer
PenDigits	9	5
OptDigits	6	10
Forest Cover	9	4
MNIST	4	20
HTRU2	16	7
SatImage-2	20	10
Credit card	12	5
Speech	3	100

3.6.4 Performance of the proposed method on single outlier type datasets

This experiment reveals the comparison of outlier detection performance of the proposed method with some of the popular methods in literature. We have shown how the use of stacked AEs enhances the performance of those methods too. Table 3.5 shows a comparative analysis for the datasets mentioned in Table 3.1.

In all the datasets, the proposed method gives the highest ROC-AUC [73]. In majority of the datasets, the F-Score [105] and G-Mean [105] of the proposed method is competitive with the best performance.

Table 3.5: Performance of different methods on single outlier type data sets

Dataset	**Performance Metrics**	**SAE-PNN (Proposed)**	**SAE-*k*NN**	**SAE-*i*Forests**	**PNN**	***k*NN**	***i*Forests**
PenDigits	ROC-AUC	**0.9923**	0.9619	0.9370	0.8880	0.9848	0.8699
	F-Score	**0.9656**	0.9563	0.7339	0.6917	0.9387	0.7268
	G-Mean	**0.9659**	0.9563	0.7517	0.7093	0.9409	0.7368
OptDigits	ROC-AUC	**0.9684**	0.9000	0.9056	0.8905	0.8467	0.6072
	F-Score	0.8951	0.9008	0.8925	0.8352	**0.9077**	0.6423
	G-Mean	0.8976	0.9008	0.8925	0.8368	**0.9101**	0.6433
Forest Cover	ROC-AUC	**0.9324**	0.8118	0.9205	0.9054	0.8453	0.7823
	F-Score	**0.9335**	0.8931	0.7377	0.9216	0.9131	0.6393
	G-Mean	**0.9335**	0.8977	0.7527	0.9244	0.9160	0.6503
MNIST	ROC-AUC	**0.9454**	0.9381	0.9414	0.8292	0.9366	0.6786
	F-Score	**0.9453**	0.9362	0.9410	0.8254	0.9350	0.6782
	G-Mean	**0.9454**	0.9372	0.9412	0.8273	0.9358	0.6784
HTRU2	ROC-AUC	**0.9262**	0.8059	0.9103	0.9216	0.8233	0.7754
	F-Score	**0.9209**	0.8723	0.9040	0.9103	0.8849	0.8015
	G-Mean	**0.9210**	0.8753	0.9040	0.9104	0.8873	0.8019
SatImage-2	ROC-AUC	**0.9589**	0.8833	0.9537	0.8661	0.9167	0.9564
	F-Score	**0.9432**	0.9372	0.7444	0.9088	0.9279	0.7247
	G-Mean	**0.9433**	0.9389	0.7630	0.9099	0.9303	0.7469
Credit card	ROC-AUC	**0.9421**	0.8821	0.9153	0.8000	0.8475	0.9008
	F-Score	0.7760	**0.9249**	0.7269	0.7896	0.9078	0.6510
	G-Mean	0.7884	**0.9260**	0.7428	0.7975	0.9101	0.6776
Speech	ROC-AUC	**0.8769**	0.6797	0.7448	0.7738	0.5400	0.4931
	F-Score	**0.8007**	0.7905	0.6644	0.6329	0.6975	0.4959
	G-Mean	**0.8038**	0.8012	0.6683	0.6437	0.7292	0.4959

3.6.5 Performance of the proposed method on multiple outlier type dataset

To validate the efficiency of the proposed method on multiple outlier type dataset, we have used four highly skewed datasets.

Table 3.6: Confusion matrix for SAE-PNN

		Predicted Class				
		Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
Actual Class	**Class 1**	1302	24	40	1	36
	Class 2	47	90	5	1	3
	Class 3	1	0	6	0	16
	Class 4	7	1	12	34	1
	Class 5	0	0	0	0	16

Table 3.7: Confusion matrix for SAE-*k*NN (*k*=35)

		Predicted Class				
		Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
Actual Class	**Class 1**	1389	5	0	1	8
	Class 2	73	73	0	0	0
	Class 3	5	0	0	1	17
	Class 4	43	1	0	11	0
	Class 5	4	0	0	0	12

Table 3.8: Confusion matrix for SAE-Random Forests

		Predicted Class				
		Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
Actual Class	**Class 1**	1388	2	0	1	12
	Class 2	77	67	0	1	1
	Class 3	6	0	1	2	14
	Class 4	35	0	0	20	0
	Class 5	5	0	0	0	11

To show the efficacy of the proposed method, we have also provided the comparison of confusion matrices obtained using each method for the Page Blocks dataset (Tables 3.6 - 3.11).

It can be seen from Figure 3.10 that the proposed method can detect multiple types of outliers as well as multiple types of inliers in the data simultaneously. It is evident from

Table 3.9: Confusion matrix for PNN

		Predicted Class				
		Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
Actual Class	**Class 1**	1043	4	277	5	74
	Class 2	48	62	12	19	5
	Class 3	0	0	10	0	13
	Class 4	0	0	8	45	2
	Class 5	0	0	0	0	16

Table 3.10: Confusion matrix for kNN (k=35)

		Predicted Class				
		Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
Actual Class	**Class 1**	1396	4	0	0	3
	Class 2	81	65	0	0	0
	Class 3	20	0	0	0	3
	Class 4	52	0	0	0	3
	Class 5	11	0	0	0	5

Table 3.11: Confusion matrix for Random Forests

		Predicted Class				
		Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
Actual Class	**Class 1**	1089	37	0	255	22
	Class 2	40	102	0	2	2
	Class 3	0	0	0	18	5
	Class 4	4	1	0	49	1
	Class 5	3	0	0	8	5

Table 3.12: Performance of different methods on multiple outlier type data set

Dataset	**Performance Metrics**	**SAE-PNN (Proposed)**	**SAE-kNN**	**SAE-Random Forests**	**PNN**	**kNN**	**Random Forests**
Statlog (Shuttle)	ROC-AUC	**0.9713**	0.8988	0.9176	0.9326	0.8988	0.8876
	F-Score	**0.9342**	0.9224	0.9317	0.8459	0.9224	0.8968
	G-Mean	**0.9349**	0.9228	0.9318	0.8496	0.9228	0.8969
Page Blocks	ROC-AUC	**0.9554**	0.8276	0.8139	0.8328	0.8239	0.8501
	F-Score	**0.6400**	0.5392	0.6226	0.6068	0.3897	0.4550
	G-Mean	**0.6414**	0.5422	0.6388	0.6107	0.3922	0.4603
ANN-Thyroid Disease	ROC-AUC	**0.9951**	0.9929	0.9921	0.8715	0.8630	0.9923
	F-Score	0.8645	0.9041	0.9075	0.6502	0.6563	**0.9464**
	G-Mean	0.8685	0.9041	0.9076	0.6548	0.6747	**0.9465**
Wine Quality White	ROC-AUC	**0.7900**	0.7172	0.7072	0.7867	0.7131	0.7050
	F-Score	0.3527	0.2246	0.3884	**0.4143**	0.2272	0.3965
	G-Mean	0.3544	0.2246	0.3917	**0.4165**	0.2272	0.3990

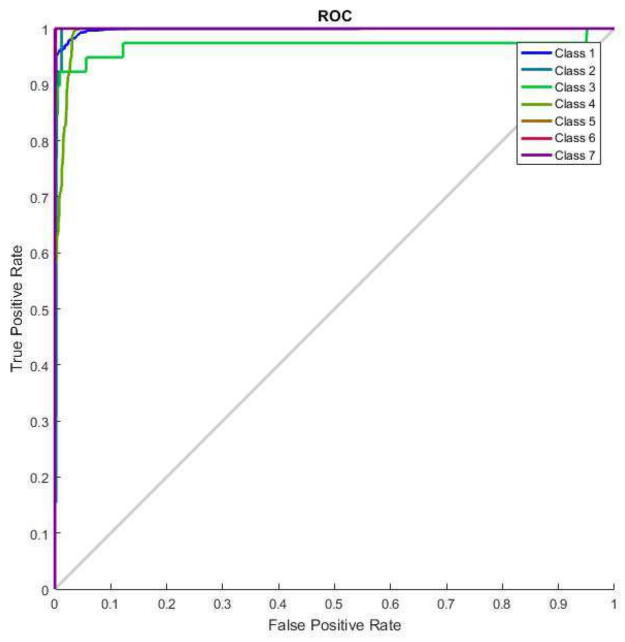


Figure 3.10: ROC plots of the 7 classes for the *Statlog (Shuttle)* by the proposed method.

Table 3.7 that most of the classifiers failed to capture Class 3 (which belongs to only 0.5% of the data). Since, for kNN, k=35 gave the highest ROC value, we have reported the results taking k=35 only for the comparison of confusion matrices. A quantitative comparison have also been provided in Table 3.12 to show the effectiveness of the proposed method in the multi-class multiple type outlier detection problem.

3.6.6 Comparison with other feature extraction methods

The above section showed how the use of AE enhanced the results of outlier detection in both single type and multiple type outlier detection tasks. However, there also needs to be a comparison on why AE was chosen as the feature extractor while there are other popular existing non-linear feature extraction methods in the literature like k-PCA, t-SNE, LLE, Isomap, GRP (refer section 2.5) etc. Table 3.13 shows a comparison between the performance of different feature extraction methods keeping the same number of PNNs used for classification for each dataset. It was seen that models built by pairing other feature extraction methods with the ensemble of PNNs mostly failed to recognise the outliers. However, use of AE was detecting outliers fairly well.

Table 3.13: ROC-AUC of different feature extraction methods paired with ensemble of PNNs on single outlier type datasets

Dataset	**SAE**	**k-PCA**	**t-SNE**	**LLE**	**Isomap**	**GRP**
PenDigits	**0.9923**	0.7909	0.9672	0.5018	0.7498	0.9322
OptDigits	**0.9684**	0.5215	0.9434	0.5021	0.5269	0.9599
ForestCover	**0.9324**	0.5	0.7396	0.5374	0.9325	0.9113
MNIST	**0.9454**	0.7513	0.8794	0.5004	0.7431	0.7987
HTRU2	**0.9262**	0.5	0.9052	0.5157	0.9194	0.916
SatImage-2	**0.9589**	0.5	0.9248	0.5576	0.9093	0.7865
Credit Card	**0.9421**	0.5661	0.7355	0.5095	0.8463	0.8672
Speech	**0.8007**	0.5185	0.748	0.5222	0.5317	0.5638

3.7 Statistical comparison of the methods

We have performed a paired t-test to compare the performance of each classifier with the other with 95% confidence level (significance level $\alpha = 0.05$ for two tailed distribution). It can be seen that use of AEs have significantly improved the performance. For any two classifiers, let c_1^i and c_2^i be the respective performance scores (in our case, the value of ROC-AUC) on the i^{th} out of the N data sets (here N=8). Then, the difference for the i^{th} dataset will be,

$$d_{12}^i = c_2^i - c_1^i. \tag{3.13}$$

Now, the average difference will be,

$$\bar{d}_{12} = \frac{1}{N}\sum_{i=1}^{N} d_{12}^i. \tag{3.14}$$

Now, the t-value for the two classifiers will be,

$$t_{12} = \frac{\bar{d}_{12}}{\sigma_{12}/\sqrt{N}}, \tag{3.15}$$

where, where σ_{12} is the standard deviation of the differences between the two classifiers over the N datasets.

Table 3.14: t-values for each pair of classifiers over the 8 single outlier type datasets.

	SAE-PNN	SAE-kNN	SAE-iForests	PNN	kNN	iForests
SAE-PNN		4.01	2.55	5.13	2.70	3.73
SAE-kNN			-2.63	-0.05	0.72	2.12
SAE-iForests				2.25	2.37	3.56
PNN					0.41	1.93
kNN						2.08
iForests						

This t-statistic will be distributed according to the t-distribution with $N - 1$ degrees of freedom (we have considered the one-tailed distribution in this case.) Since, we are comparing over 8 datasets (i.e. $N = 8$), our degrees of freedom will be $8 - 1 = 7$. (We have omitted the results for the multiple outlier type datasets since the method iForests was replaced with Random Forests in that case.) The corresponding p-values are provided in Table 3.15. Taking the Bonferroni [106] correction into consideration, we get the Bonferroni corrected critical p-values as

$$p_{critical}^{corrected} = 1 - (1 - p_{critical}/n_t)^{n_t}, \tag{3.16}$$

where, n_t denotes the number of comparison tests made. In our case, since fifteen comparisons were made, $n_t = 15$. If we set our critical value of p to $p_{critical} = 0.025$, (which corresponds to a 95% confidence interval for one-tailed tests) then the Bonferroni corrected critical value will be $p_{critical}^{corrected} = 0.025/15 = 0.024710429$.

If the p-value for the two classifiers p_{12} is greater than the critical value $p_{critical}^{corrected} = 0.024710429$, we reject the null hypothesis. We can see from Table 3.16 that the use of stacked AEs clearly enhanced the performance to a significant level. As both the methods kNN and PNN have identical hypotheses, the performance of a PNN classifier and a kNN classifier is comparable. Using SAE with iForests gave better results than using iForests

Table 3.15: p-values for each pair of classifiers over the 8 single outlier type datasets.

	SAE-PNN	SAE-kNN	SAE-iForests	PNN	kNN	iForests
SAE-PNN		0.002553	0.019095	0.000679	0.015268	0.003677
SAE-kNN			0.016995	0.481958	0.246100	0.035730
SAE-iForests				0.029731	0.024950	0.004629
PNN					0.347259	0.047567
kNN						0.037999
iForests						

Table 3.16: Significance results for 95% confidence (here 1 means significant and 0 means non-significant)

	SAE-PNN	SAE-kNN	SAE-iForests	PNN	kNN	iForests
SAE-PNN		1	1	1	1	1
SAE-kNN			1	0	0	0
SAE-iForests				0	0	1
PNN					0	0
kNN						0
iForests						

alone.

However, using stacked AEs with kNN did not give a significant improvement over kNN. Classification with kNN in the original feature space proved out to be comparable with the classification in the space of features extracted by SAE. Interestingly though, SAE-kNN could outperform SAE-iForest where kNN and iForests were comparable. Using an ensemble of PNNs in the feature space extracted by the SAE proved to be the best. The proposed method outperformed every other method. We can therefore say that non-linear transformations given by the AEs are helpful for highly imbalanced datasets for outlier detection.

3.8 Conclusion

In this chapter, we have proposed a supervised setting of outlier detection using deep learning in both single type outlier as well as multiple outlier type datasets. We have used stacked AEs as feature extractors and then have used a set of PNNs to classify the outliers and inliers in the data. Such a system is shown to perform better than the existing methods for both multi-class single type outlier detection as well as multi-class multiple type outlier detection problems.

As we are dealing with supervised learning models, in order to have a fair comparison, we only considered methods that use outlier information as well. We would like to explore support vector data description (SVDD) for multi-class multiple type outlier detection scenario in future studies. The proposed method of multiple outlier type detection may be extended in the direction of unsupervised methods too, which currently put every new outlier to a single outlier class only.

We can conclude that the use of stacked AEs clearly enhances the performance of all the methods. However, the proposed method outperforms all other methods. Experiments have also revealed that going deep (or using more hidden layers) in the feature extraction stage provided an enhancement of performance of the proposed method. But, too much depth degrades the performance again.

Chapter 4

Deep Feature Extraction for Change Detection using Unsupervised and Active Learning

4.1 Convolutional Autoencoder

Convolutional autoencoders (CAE) are another types of AEs which are efficient for image data. They have the same task as an ordinary AE: to reconstruct the input. In chapter 2, the concept of a CAE is already described. Usually in the middle code layer, a standard undercomplete convolutional autoencoder (CAE) would have the maximum number of filters and the size of the image will be reduced. If there are multiple (more than two) layers of non-linearity (in this case convolution layers are the non-linear operations), then the CAE is termed as a deep convolutional autoencoder (DCAE). This is depicted in Figure 4.1. The number of filters in each layer successively increase in the encoder part till the middle layer, and then the number of filters in each layer successively decrease in the decoder part. This bottleneck allows the network to extract some interesting features which are very compact descriptors of the input image. To understand the efficiency

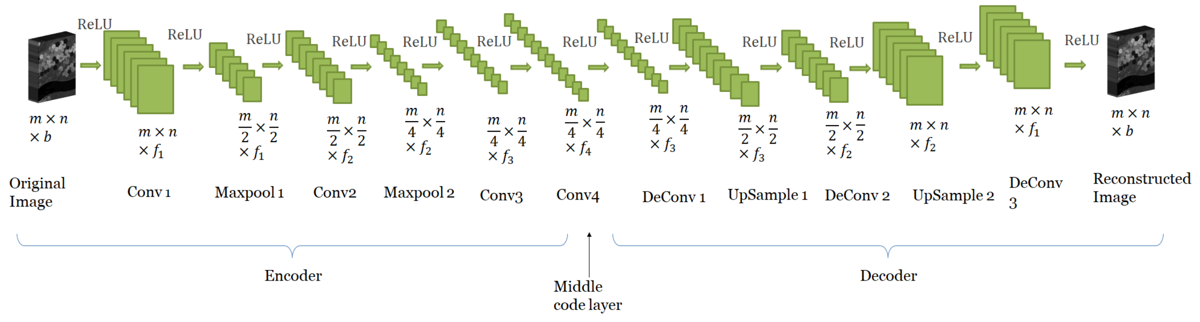

ReLU
$m \times n \times b$
Original Image
$m \times n \times f_1$
Conv 1
$\frac{m}{2} \times \frac{n}{2} \times f_1$
Maxpool 1
$\frac{m}{2} \times \frac{n}{2} \times f_2$
Conv2
$\frac{m}{4} \times \frac{n}{4} \times f_2$
Maxpool 2
$\frac{m}{4} \times \frac{n}{4} \times f_3$
Conv3
$\frac{m}{4} \times \frac{n}{4} \times f_4$
Conv4
$\frac{m}{4} \times \frac{n}{4} \times f_3$
DeConv 1
$\frac{m}{2} \times \frac{n}{2} \times f_3$
UpSample 1
$\frac{m}{2} \times \frac{n}{2} \times f_2$
DeConv 2
$m \times n \times f_2$
UpSample 2
$m \times n \times f_1$
DeConv 3
$m \times n \times b$
Reconstructed Image
Encoder
Middle code layer
Decoder

Figure 4.1: Architecture of a standard deep convolutional autoencoder.

of a DCAE in supervised setting, we have explored the idea of trigger detection as an extension to supervised learning of hand gestures in the American sign language.

4.1.1 Application as a Feature Extractor in Trigger Detection

Recently the technique of automatic speech recognition (ASR) [107] has gained significant reputation as an important part of artificial intelligence. As one of the most prominent applications, automatic trigger-word (also known as wake-up-word) recognition has developed. These methods are also known as automated voice trigger systems. Such systems play an important role in human-computer interaction. These systems act as a contactless switch instead of the traditional push-to-talk mode and appropriately fit the recent multitasking lifestyle.

However, these methods being voice activated are inefficient for people who are incapable of speech or are in some silence zone. Touch based trigger gesture detection has been developed and can be seen to be used in mobile phones, laptops, etc. However, contactless trigger gesture detection still lacks a substantial research. Therefore, a computer vision based contactless switch should be developed so that it could be used universally without any difficulties. Recognition of hand gestures can be done in two ways. People use electromagnetic gloves and sensors which capture the features like hand shape, movements and orientation of the hand and try to figure out the shape. However, such method is not cost effective as people cannot always afford the gloves or the sensors. Therefore, researchers from computer vision have developed gesture recognition systems, which involves image processing tasks [108]. Researchers have recently diverted their attention to machine learning for such gesture recognition tasks [109]. However, the literature lacks a development towards a trigger detection system using such gesture recognition techniques.

We have developed a trigger detection system [49] for the ASL by making use of Convolutional Autoencoder pre-initialisation. By trigger detection, we aim at building a customisable contactless switch that can turn 'on' if it finds a trigger gesture in any video that it receives and stays 'off' if it does not. For simplicity, we have considered the well known American Sign Language (ASL) [110] for experimental purposes. It is a complete and complex language, in which people use hand signs to communicate. It is the fourth most commonly used sign language in the world. This language has 26 alphabetical signs, two of which (alphabets J and Z) are motion based gestures. A sign language based MNIST dataset was developed in [111] that consists of images for the 24 static hand gestures (alphabets J and Z were excluded as they are motion gestures).

Developing voice activated trigger systems [112, 113] is an extensively researched topic in the literature. Researchers have been using Hidden Markov Models (HMM) [114] for such speech recognition tasks. Recent trends witness use of deep neural networks (DNN) [115] for such trigger-word recognition. Recurrent neural networks (RNNs) are also being used [116] for such wake-up-word or trigger-word detection in speech. However, such voice activated systems are often useless for deaf and mute people. Since, such people use sign languages to communicate, there have been many attempts to automatically recognise the gestures [108, 109, 117, 118]. For static hand gesture recognition in real time, we need models which have real time testing phase. Support Vector Machine (SVM) models [109, 117] are quite efficient, but, they are a bit slow as compared to real time deployment.

In [109], they have extracted HOG features from the image and used an SVM to classify the gestures. [119] has examined various classifiers like C4.5, Random Tree, Naive Bayes, Random Forest, SVM (linear kernel), SVM (RBF kernel) and Artificial Neural Network (ANN-Multilayer Perceptron) in context of gesture recognition. It was seen that Random Forest, SVM and ANN performed better than others. Using RBF kernel in SVM did not produce any added efficiency over linear kernel, but increased the time complexity.

Deep neural networks have also been used to recognise sign languages [120, 121, 122]. However, such methods use spatial-temporal features from raw videos in the training set and are much more complex to be used for small scale trigger applications. We have developed a much simpler trigger detection system that uses convolutional neural networks (CNN). In the training phase, we have used images of all the 24 static hand signs and trained our CNN model on these images. During the testing phase, each video frame captured in real time was analysed and if the trigger image was present anywhere in the video, the system would trigger. Our model is efficient as it does not run through the remaining video frames once a trigger is detected. The switch will be turned 'on' only on detecting the trigger and otherwise will remain dormant.

To the best knowledge of the authors, the gesture recognition problem was not extended to any kind of trigger detection scenario. It was therefore the only choice to compare out method with the existing gesture recognition methods in literature like SVM [117], Random Forests, ANN [119] and HOG+SVM [109].

4.1.2 Convolutional Autoencoder for Pre-training

The proposed method makes use of deep convolutional neural network (Deep CNN) for gesture recognition. We argue that using CNN was essential to capture the complexity of the hand signs as most of them look similar. Figure 4.2 [111] shows the various gestures and the alphabets they represent. It is evident that most of the signs like 'm' and 'n' are very similar.

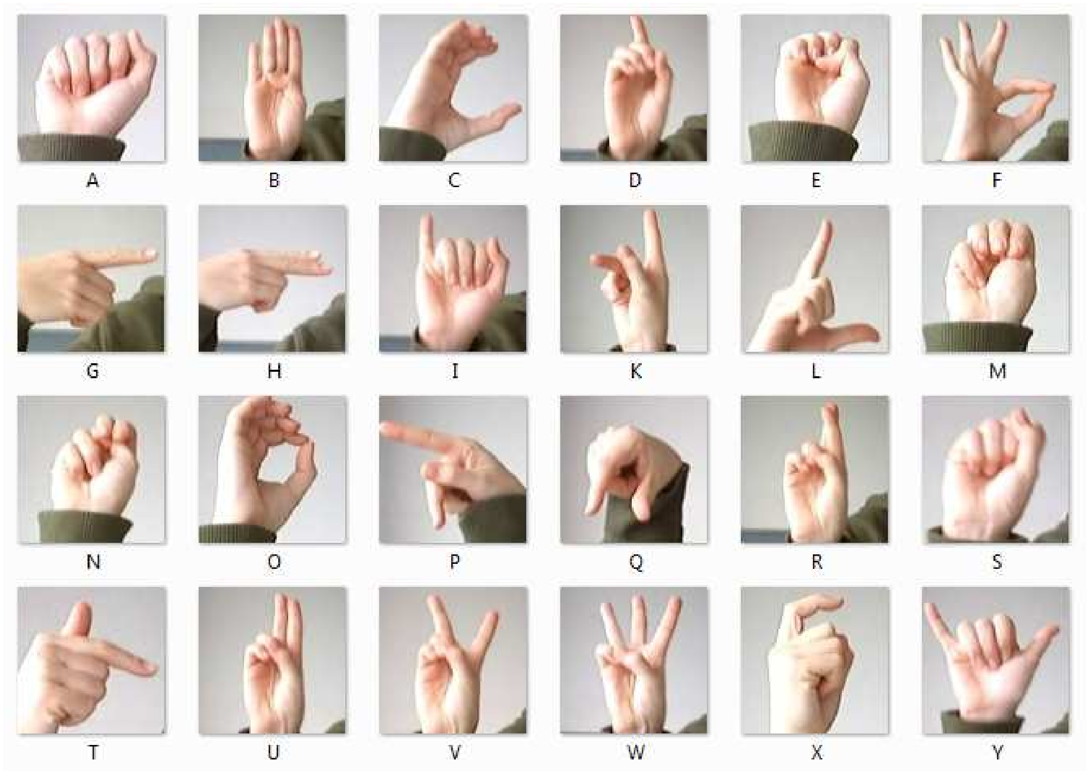


Figure 4.2: 24 Static Hand Gestures for American Sign Language Letters

The similarity among the various hand gestures make it a difficult problem to clearly differentiate them. CNNs can capture the variations in the different signs and therefore is opted for most of the computer vision tasks [15]. It is an advantageous property of CNN to automatically extract features at each hidden layer and hence make it very efficient for raw data processing tasks. Unlike other computer vision gesture recognition systems CNN does not need any feature engineering. Therefore, using a CNN was an inevitable choice for the contactless gesture trigger detection problem. To do so, we have extended a conventional multi-class gesture recognition problem to a trigger detection problem.

We have used a convolutional auto-encoder network for pre-training purposes. The architecture of the convolutional auto-encoder network is shown in Figure 4.3. A convolutional auto-encoder learns to reconstruct the input and thus learns a transformed feature space which is capable of reproducing the input approximately. A typical CAE combines the advantages of a CNN as well as a normal auto-encoder (AE). It is known

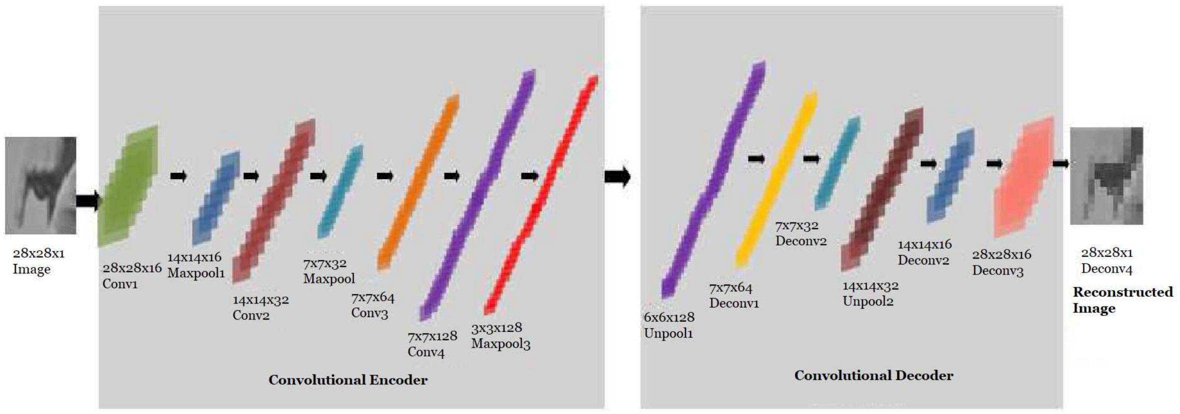


Figure 4.3: Architecture of the Convolutional Auto-Encoder Network used for Pre-Training Purposes

that unsupervised pre-training enhances the performance of a multi-layer neural network by setting the random weights to a stricter region of weight space and hence the supervised fine tuning finds the good weights easily. In contrast to the fully connected topology for AE, CAE is made of an encoder which contains convolutional and maxpooling layers and a decoder which contains deconvolutional and unpooling (or upsampling) layers.

They are trained to recreate the input at the output nodes, and are thus used as feature extractors or as a pre-trained model for the final convolutional network.

4.1.3 Supervised Classifier

The encoder of the CAE is stacked on top of a fully connected network with 24 output nodes.

We have used four convolutional layers and three maxpool layers in the Deep CNN model. Finally the model was re-trained on the training sample labels to minimise the training loss. The architecture of the model is shown in Figure 4.4.

A schematic diagram of the proposed trigger detection system has been shown in 4.5. The CNN learns all the 24 gestures during this training phase. During the test phase, we add a customising module that allows the user to select his/ her preferred gesture for

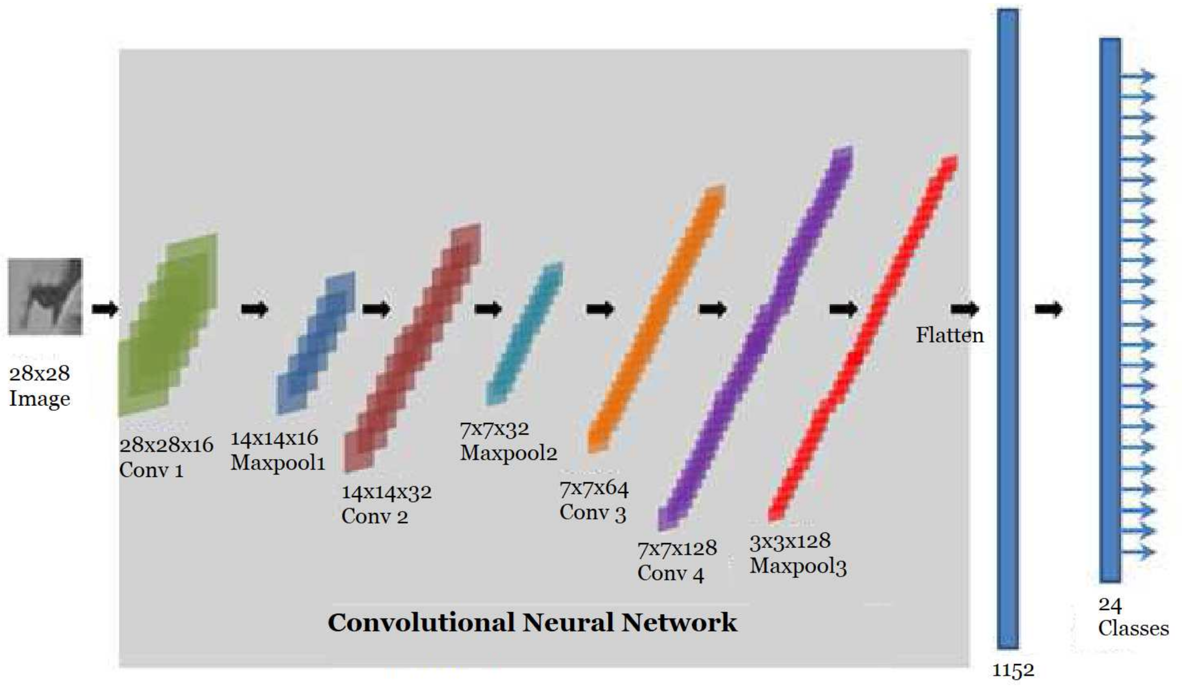


Figure 4.4: Architecture of the Deep CNN Model used for Gesture Classification

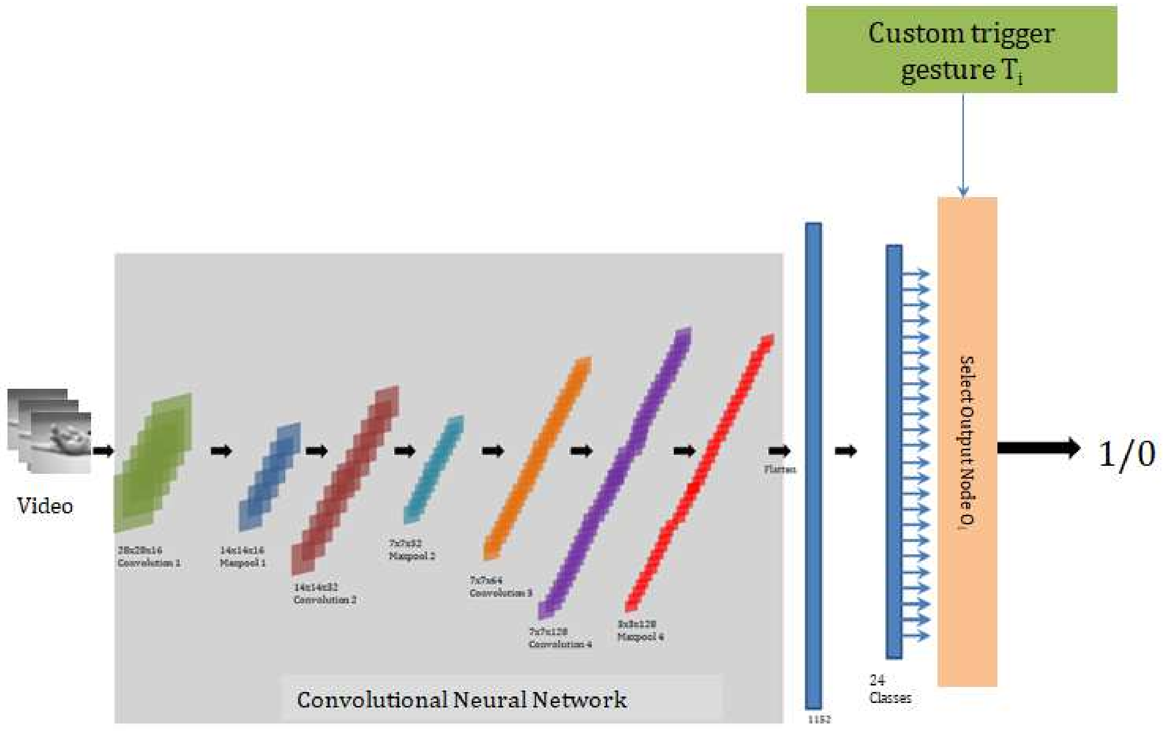


Figure 4.5: Architecture of the Total Model used for Custom Trigger Gesture Detection

Table 4.1: Comparison of the Accuracy of Trigger Detection for each Trigger Gesture T_i

Trigger Alphabet T_i	**Proposed**	**HOG + SVM (Linear Kernel)**	**Random Forest**	**SVM (Linear Kernel)**	**MLP (2 Hidden Layers)**
A	**1.0000**	**1.0000**	0.9819	**1.0000**	**1.0000**
B	0.9606	**0.9884**	0.8194	0.9005	0.8449
C	0.8677	**1.0000**	0.9258	0.9839	0.9710
D	**0.9224**	0.9184	0.8327	0.9592	0.8816
E	**0.9940**	0.9820	0.8815	0.9659	0.9659
F	**1.0000**	**1.0000**	0.8947	0.9150	0.9150
G	**0.9397**	0.9310	0.7845	0.8994	0.8276
H	**0.9977**	0.9495	0.8073	0.8303	0.8440
I	**1.0000**	0.8646	0.4653	0.9028	0.6667
K	**0.9305**	0.8731	0.4743	0.4773	0.5347
L	**1.0000**	0.8134	0.8612	0.8421	1.0000
M	**0.9467**	0.9086	0.5533	0.7487	0.6726
N	**0.9794**	0.7491	0.3952	0.6942	0.5704
O	**0.9228**	0.9146	0.6585	0.7398	0.8943
P	**1.0000**	0.9625	0.9308	0.9395	1.0000
Q	**1.0000**	0.8720	0.9573	0.9024	0.7439
R	**0.9514**	0.8542	0.4375	0.7153	0.5694
S	0.9106	**0.9146**	0.3862	0.6829	0.4634
T	**0.8347**	0.7500	0.5968	0.6734	0.6331
U	**0.9887**	0.8759	0.3383	0.6241	0.4586
V	0.8353	**0.9364**	0.3237	0.7139	0.5000
W	**1.0000**	0.8981	0.4126	0.8058	0.8058
X	**1.0000**	0.9326	0.5281	0.6217	0.6966
Y	0.8795	**0.8825**	0.4910	0.6205	0.7048

trigger. The model is then allowed to check through all the frames in the video.

If any frame has the trigger gesture T_i, the node O_i at the output layer will have the maximum response. Then the model will automatically switch to the 'on' or '1' state from its dormant '0' state and will not check any further frames in that video. Suppose there are N_j frames $\{F_1, F_2, ...F_{N_j}\}$ in the video V_j (where j=1, 2, ..., 7000). If a frame F_k is fed forward through the CNN, it will produce response at the output nodes as $O = \{O_1, O_2, ..., O_{24}\}$.

4.1.4 Results

To establish the efficacy of such a model, we have compared our model with four other state of the art methods. We have given a complete view of how our model performs for different gestures selected as the trigger gesture by the user. A comparative analysis of the various methods have been shown in Table 4.1.

It is evident that most of the other methods perform poorly for some of the confusing signs like 'M' and 'N' and 'G' and 'H'. Our method can clearly capture the complex difference between the gestures except for some where HOG+SVM performs better. However, in those cases too, the proposed method is competitive to HOG+SVM as the difference in performance is not too pronounced except for the letter 'V' and 'C'.

4.2 Convolutional Autoencoders as Unsupervised Image Feature Extractors

In the above section, it was established that CAEs could clearly capture the variations in the different signs which look similar to human eye and thus, are excellent feature extractors for images. The above experiment was performed to see the capabilities of a CAE for supervised image classification. However, in cases of unsupervised tasks, the potency of CAEs need to be explored. If the CAEs have more than two convolution layers, they may be termed as deep convolutional autoencoders (DCAE) as they have multiple layers of non-linearity.

To prove that CAEs are good feature extractors, we need to concentrate on more complex images like hyperspectral images and more complicated tasks like change detection. Therefore, this has motivated us to extend the use of CAEs to Hyperspectral Images in the following parts. Binary change detection in bi-temporal co-registered hyperspectral images is a challenging task due to the large number of spectral bands present in the data.

Researchers, therefore, try to handle it by reducing dimensions. The following work aims to build a binary change detection system for a pair of such bi-temporal co-registered hyperspectral images using the feature extraction property of DCAE.

4.3 Deep Convolutional Autoencoder for Unlabelled Hyperspectral Images

A remotely sensed hyperspectral image (HSI) [123] is a complex image data which has multiple bands (narrowly spaced). Each band of a remotely sensed hyperspectral image of a particular region captures multiple wavelengths of the electromagnetic spectrum. Hyperspectral imaging seeks to utilize the spectral information which has more information about the region than multi-band images. Hyperspectral images are basically, image cubes that consist of hundreds of spatial images. Each spatial image, or spectral band, records the responses of ground objects at a different wavelength. For example, the NASA AVIRIS (Airborne Visible/Infrared Imaging Spectrometer) captures the spectral images in 224 contiguous spectral bands at the visible and near-infrared regions. The spectral range in hyperspectral data extends beyond the visible range (ultraviolet, infrared). All these images may not be visible to the human eye, but contain important information to identify ground objects. Hyperspectral images helps us to explore the unexplored, by visualizing information invisible to human eyes.

However, since it contains multiple bands, sometimes the information about different changes is often found in different bands. That is why, it becomes a complicated task to detect changes between two registered hyperspectral images of the same place when there are diverse minute and bigger changes. To do the same in an unsupervised manner is even more challenging. Researchers have proposed a wide range of such digital change detection methods ranging from supervised, semi supervised and unsupervised methods.

Supervised methods [124] include training the system with some training labels from the ground truth data. A majority of the pixels (usually 70 percent) are labelled as changed or unchanged. The classifier system tries to learn the changed and unchanged classes through the labels provided. It then detects which of the remaining pixels are changed or unchanged. This kind of supervised change detection may be regarded as a problem of classifying imbalanced classes (as the number of unchanged pixels are more than number of changed pixels). Semi-supervised methods [60, 125] provide limited training data to the classifier system. There are only a few labelled samples (mostly from the unchanged class). Researchers have also explored different selective labelling techniques like active learning which share resemblance to the semi-supervised methods. Manual labelling of pixels are quite cumbersome and expensive. Moreover for HSIs, the labelling becomes challenging due to the sheer number of spectral bands. Since unsupervised methods [57], mainly rely on detecting the points of dissimilarities between the two images, they do not require any manual labelling. Due to this reason, unsupervised methods are mostly less accurate than the supervised [51, 56, 59] or semi-supervised [60, 61, 62] methods but are much more sophisticated in application.

In this chapter, an unsupervised change detection system is proposed that makes use of feature extraction by a novel feature fusion convolutional autoencoders (FFCAE). The traditional convolutional autoencoders is modified to suit the feature extraction process for change detection in hyperspectral images. The proposed feature extraction technique addresses three main concerns that arise during change detection in bi-temporal co-registered HSIs:

(i) Given two HSI, lower their dimensionality in the most efficient way possible.

(ii) Detection of minute changes with lesser misses and false alarms.

(iii) No manual labelling required.

Keeping the problem statement in mind, the proposed solution presents a lower dimensional feature map which is enough to detect the changes between the two co-registered bi-temporal hyperspectral images. The feature extracting FFCAE network has multiple hidden layers in the encoder as well as the decoder part unlike a stacked version of autoencoder. It is different from a standard deep convolutional autoencoder (DCAE) as it preserves both the spatial as well as the spectral information in its hidden code layer along successive levels and multiple receptive fields. Results show that the proposed method outperforms all the other methods it has been compared with.

The subsequent section 4.3.1 describe the literature survey on the present topic. Section 4.3.2 provides a detailed overview of the proposed methodology and section 4.3.4 reports the results obtained. Finally, section 4.5 concludes the chapter and also suggests the directions of future works.

4.3.1 Literature Survey

Most change detection algorithms for HSIs [51, 59, 60, 126] can detect major changes i.e. major land transformations but they are not able to detect subtle changes, which appear only in few spectral bands like water content, crop growth differences, land cover transitions etc. Noise generated by both external sources (atmospheric effects due to absorption and scattering) and internal sources (instrument noise) also affect the hyperspectral sensors. For this reason, need of robust methods of change detection is necessary.

Supervised models classify the pixels of each single image from different time periods using supervised classifiers and then compare the generated maps to detect changes. Recent supervised methods which show promising performance are using convolutional neural networks [51, 56, 59] for classification of pixels, but this requires the ground truth. Manually labelling ground reference data is expensive and unfavourably low. This is why, semi-supervised [60, 61, 62] or unsupervised [127] methods of change detection are favoured which do not require extensive efforts of human annotation. A recent semi-

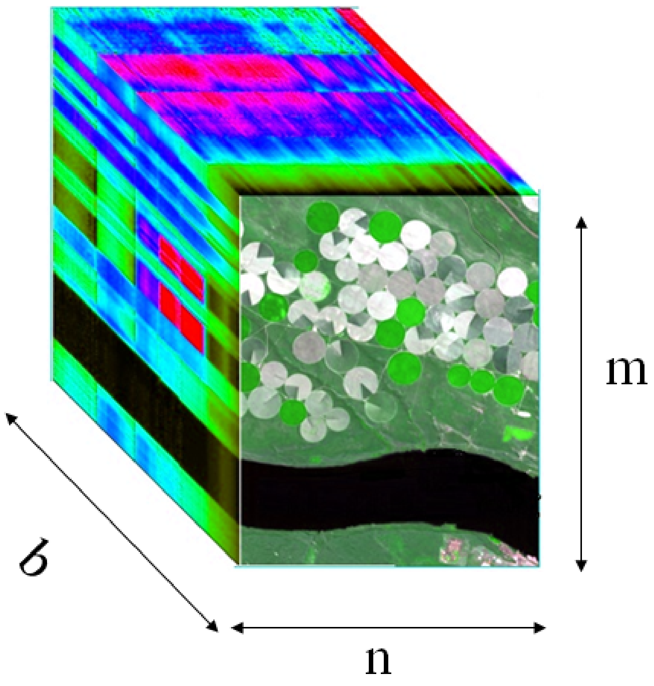


Figure 4.6: A conceptual diagram of a hyperspectral image.

supervised method of change detection in hyperspectral images [60] involves the use of pseudo-labels. However, such methods are again dependent on human labelling of unchanged pixels. To avoid the expenses of gathering ground truth data, unsupervised change detection methods are much more gracious. This chapter, therefore primarily concentrates with the domain of unsupervised change detection only.

Traditional unsupervised solutions include finding the difference image (DI) and then thresholding it or clustering the pixels. Let a hyperspectral image **I** be of size $m \times n$ and has b number of bands as shown in Figure 4.6. So, each pixel of image **I**, denoted as $\mathbf{p}_{ij}$ will be a vector of b dimensions.

$$\mathbf{p}_{ij} = \{p_{ij}^1, p_{ij}^2, p_{ij}^3, ..., p_{ij}^b\}, \tag{4.1}$$

where, $i = 1, 2, 3, ..., m$ and $j = 1, 2, 3, ..., n$.

Let us consider a pair of bi-temporal co-registered hyperspectral images (with b bands)

I_1 and I_2 of the same location photographed over different times. The assumption of them being co-registered fulfil that for all the pixels in the two images, pixel $\mathbf{p}^1_{ij}$ for I_1 and pixel $\mathbf{p}^2_{ij}$ for I_2 represent the corresponding point of the original location. The change occurred over the location can be seen by generating a binary change map C such that the ij^{th} pixel c_{ij} of C is marked as changed (by setting the value to one) if the location captured by the ij^{th} pixels for both images has changed and is marked as unchanged (by setting the value to zero) if the location has not changed. The generation of change map may be formally constructed by the equation,

$$C = g(f(I_1) \ominus f(I_2)), \tag{4.2}$$

where, $f()$ is the feature transformation function, $g()$ is the decision function and $\ominus$ is the difference operator.

Simple image differencing technique [128] just subtracts the two images pixel by pixel. Some techniques include generation of a change vector [129, 130]. The change vector is a $m \times n$ image with all the spectral information is aggregated into a single plane. Another method is image ratioing. However, as the ratio is not a normal distribution, researchers have criticised its usage in change detection methods. Researchers [131] therefore use a log ratio of images.

Without feature transformation, one may often end up struggling with noise which are falsely detected as changes [57]. Feature transformation model the relationship between the image object and its real-world geographical feature as closely as possible by reducing the amount of such false alarms and missed changes. Considering only the spectral pixel level information often results in high false alarms [57]. For this reason, many methods incorporate pixel level information as well as spatial-context information. Researchers have proposed many methods to handle this problem. In the line of linear feature transformation methods, a PCA based technique is proposed [132], which employs an advanced technique by doing a window based PCA technique for hyperspectral images. This incorporates both the spatial as well as spectral information. However, a linear transformation is often not enough to represent the complex relationships between the two images. So, researchers have proposed non-linear methods like iteratively re-weighted multivariate alteration detection (IRMAD) [133] which has been claimed to be better at dimensionality reduction of hyperspectral images for change detection. This method has recently been supplemented with synthetic fusion of images [134] called SFI-IRMAD which has shown to give a boost to the change detection performance. Another recent method showed promising results for change detection in hyperspectral images using synthetic fusion of images [58]. In this case, they have used a custom difference operator that is mainly based on the spectral angle mapper (SAM) [135]. With the popularity of deep learning, some deep convolutional networks have also been proposed [56, 126] for change detection.

A method called S3DCAE [56] uses a three dimensional deep convolutional autoencoder to extract relevant features from the hyperspectral images. However, the results reported make use of limited labelled samples and falls under the domain of semi-supervised change detection. They claim that their method of feature extraction is robust and thus the extracted features may easily be used for unsupervised change detection as well. Another unsupervised deep convolutional autoencoder based method called DSFANet [126] is proposed which showed good change detection capabilities. This method uses deep feature extraction of the two images and then does a slow feature analysis on the absolute difference image generated by the two features. It is quite obvious that all of these methods are basically reducing the dimensions of the two hyperspectral images and have mostly used k-means as the final decision function $g()$ on the difference image generated. All these methods somehow try to reduce the missed changes and false changes as much as possible. Any method which provides a better balance will be deemed superior. Thus, the present chapter proposes a novel unsupervised method of feature extraction that is capable of detecting changes with a better performance than the above state of the art methods. The details of the proposed architecture is provided in the next section.

4.3.2 Proposed Methodology

The proposed methodology utilises an unsupervised neural network to train on a pair of bi-temporal co-registered [136] images of a particular region. The main motivation is that, the neural network learns to compress all the band information into a transformed image of lower dimension, which would retain maximum possible information from all. The description of the datasets is provided in the next section.

4.3.2.1 Dataset Description

Each of the datasets taken in the study are a pair of bi-temporal hyperspectral co-registered [136] images. We have taken four pairs of hyperspectral images for the study. The first pair of images, named as USA dataset, corresponds to an irrigated agricultural land in the Hermiston city of USA. The two hyperspectral images, each with a size of 307×241 pixels, were acquired by the Hyperion sensors respectively on May 1, 2004, and May 8, 2007 [137]. The second pair of images are, named as the China dataset, corresponds to a farmland in the Jiangsu province of China. These images were acquired by the Hyperion sensors May 3, 2006, and April 23, 2007, respectively [137] and have a size of 420×140 pixels. Both the USA dataset and the China dataset images consist of 154 spectral bands. The third dataset in consideration, named as River dataset, is of the Yangtze river in the Jiangsu Province of China [59]. The two images were captured by the Hyperion sensors on May 3, 2013, and December 31, 2013, respectively. The River dataset has a size of 463×241 pixels and has 198 spectral bands. It is to be noted that although the intrinsic number of bands for the Hyperion sensors is 242, the above datasets available freely had lesser number of bands as they were already subjected to the band removal process and only the bands with high signal to noise ratio were available publicly [59]. We have not used any kind of pre-processing on the available images and have considered them in our study in the same form as they were available. The Hermiston dataset [138] was aquired by the Hyperion Sensors in the years 2004 and 2007 respectively over the Hermiston City area in Oregon. The image sizes are 390 × 200 each and includes 242 spectral bands. Out of these 242 bands, only 198 channels were informative. The rest of the 44 channels had the same value in all the pixels and hence had zero entropy (i.e. no information). So, the 44 channels with zero-entropy were omitted and only the 198 channels were considered for further processing. Although, the ground truth showed 5 types of changes related with crop transitions, we converted this to a binary change vs no-change problem and all the 5 change classes were merged to the single change class.

The next subsection describes the particular type of autoencoder configuration that has been used.

4.3.2.2 Proposed Feature Fusion Convolutional Autoencoder (FFCAE)

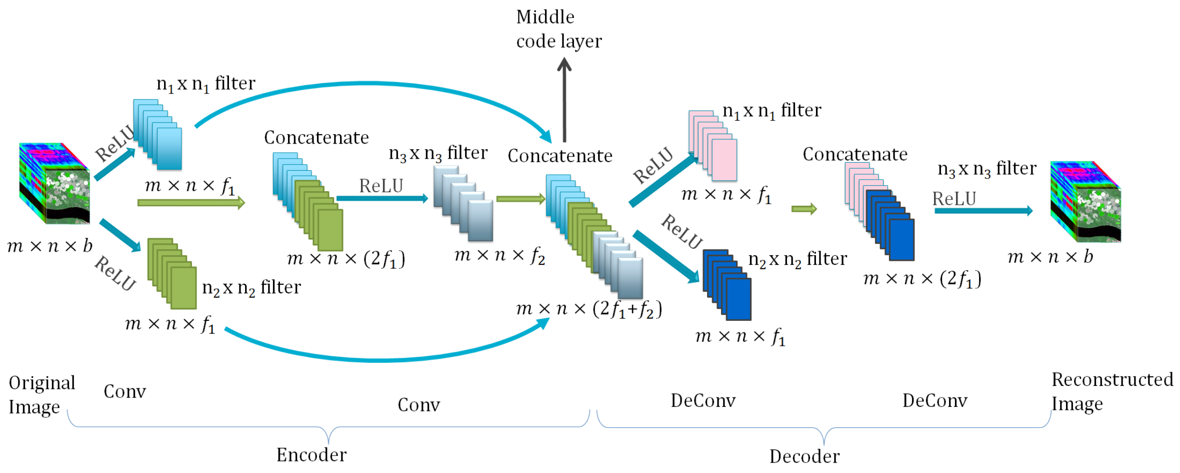


Figure 4.7: Architecture of the proposed FFCAE (blue arrows represent the trainable weights).

The main challenge with the afore-mentioned hyperspectral images are that they have a huge number of bands, each of which contain a great deal of information about the topology of the region. Moreover, the dimensions of these images are usually of the order of hundreds or thousands. So, to use a supervised convolutional network (CNN) would become very time consuming. On the other hand, AEs would be quite efficient

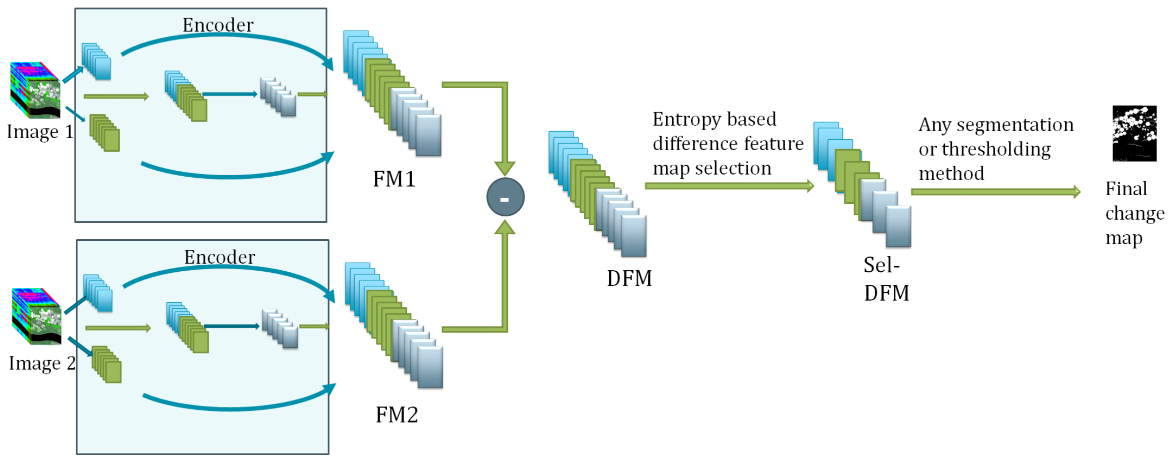

Encoder
Image 1
FM1
Encoder
Image 2
FM2
-
DFM
Entropy based difference feature map selection
Sel-DFM
Any segmentation or thresholding method
Final change map

Figure 4.8: Overall procedure of change detection.

in reducing the dimensions (both in bands as well as in size) of these images. AEs map different values into different ranges [16, 17]. Noise are treated differently than changes in an AE. An undercomplete AE passes the input through a bottleneck which discards the sporadic values (as observed for noise) while learning informative features that is used to reconstruct the input back. The majority of the patterns present in the data (here unchanged pixels) get mapped to similar value ranges while the changed pixel values will be revealed as they get mapped to a different range. Thus, it is intuitive that AEs can be used to reduce dimensionality of HSIs because if a pixel has changed its value between two images, the two values will be mapped to different ranges.

CAEs are good for image data as they take both spectral as well as spatial information into consideration. A standard undercomplete convolutional autoencoder would alter the size of the images using pooling and upsampling operation as well as change the number of filters through the convolutional and deconvolutional operations. They are primarily equipped with ReLU (rectified linear units) as activation along the layers.

The proposed architecture [Figure 4.7] is different from any traditional convolutional autoencoder architecture. The novelty of the proposed architecture is that it has exploited the possibility of playing with the size as well as the number of filters in a convolutional autoencoder and added skip connections to create a fusion of lower level and higher level features. In the proposed architecture, the original image is convolved with filters of two different sizes successively. This results in the generation of lower level features with different receptive fields. The features are then concatenated and is convolved with another filter to generate a higher level feature representation. Then both the lower level features and the higher level features are concatenated with a skip connection at the next level (which is the middle code layer). This is the part where feature fusion is taking place. The decoder then convolves the fused feature with filters of two different sizes again and then the concatenated feature maps is convolved with another filter to reconstruct the original image back.

Choosing the window size and the number of filters in a CAE is crucial because the entire physics behind the feature extraction property of a CAE depends on these two parameters mainly. Usually the filter windows in a CAE are taken to be odd i.e. 1×1, 3×3, 5×5, 7×7, etc depending on the type of image (1×1 is usually not taken as it implies no spatial information is considered). The windows 1×1 and 3×3 are considered to be smaller size and 5×5, 7×7, etc are considered as larger sizes. If the window size is taken smaller, then it may miss some spatial contextual information but as the number of parameters is less, it would be less prone to overfitting. On the other hand, larger window size might cause overfitting but can capture spatial information well. We can combine these two effects by using a multi-window approach during feature extraction. Window size in a CAE is equivalent to the receptive field (i.e. how much of the image the filter sees at a time). Effect of larger receptive field can be achieved by increasing depth. However, going too deep is also problematic for training sometimes. Depth captures higher level features which imply strong object level semantic information and the first layer extracts lower level features which have acute discriminative capabilities. However, there needs to be a balance. Acute discriminative features get influenced by noise while strong object level features often ignore tiny changes. Thus, a fusion of higher level and lower level features may create this balance.

In the proposed architecture, twin filters are used (3×3 and 5×5) i.e. $n_1 = 3$ and $n_2 = 5$. Adding another layer extracted higher level features, and inherently increased the receptive field. We chose $n_3 = 3$ as it gave the effect of a 5×5 and 7×7 receptive field. Increasing the depth would have increased the receptive field even more, but that would incorporate additional burden of increased number of parameters. Using two filters ensure that both the spatial information as well as subtle changes are captured. The fusion of lower level features and higher level features is obtained through a skip connection. The code-layer therefore produces the most compact transformed image which would have both the spatial as well as spectral information of the change. It must be noted that even after passing through the layers of the network, the image size is not reduced below its actual size anywhere. This was purposely done to preserve the resolution in the extracted feature maps. After the FFCAE network is trained to reconstruct the input back, the encoder part may be used to extract the deep feature maps (DFM) from the middle code layer.

The proposed FFCAE being an extension of CAEs is structurally built to extract features to reconstruct the input. Since there is no label information provided to the network, it captures some common generic features shared by both the images. However, these features are mapped in different feature maps than the discriminating ones. We can, therefore, easily discard such feature maps which are the same for both images using a simple entropy based filtering i.e. reject the feature maps for which,

$$\epsilon(f(I_1) \ominus f(I_2)) = 0, \tag{4.3}$$

where, $\epsilon()$ is the entropy of an image. The feature maps which are selected (Sel-DFM) are used for further change analysis. The overall procedure of change detection is shown in Figure 4.8.

The trick is that, the same FFCAE was trained with only two images for 50 epochs. It is done with the intuition that most of the pixels in the two images are almost identical as they have no change. So, it would be much more gracious to train a single network to reconstruct the two images.

4.3.3 Contributions

The proposed method handles the major challenges faced in hyperspectral change detection problems which are under-detection and over-detection of changed pixels. They are usually caused due to similarity in the intensity values of the changed and unchanged regions, faulty classification by colour thresholding or texture features. If there are many types of changes occurring over long periods of time, then this makes the task much more complicated. The changes are usually distinguished by their characteristic spectral signatures and thus, some minute changes might often get overshadowed by a prominent large change. The proposed method thus adds two major contributions to the change detection task:

i light-weight feature extraction which maps discriminating features into separate feature maps, and

ii detection of both minute and major changes simultaneously without any label information.

4.3.4 Results

The proposed method was implemented and compared with six state-of-the art change detection methods. In order to have a fair comparison, the methods which use any kind of labelling (supervised, semi-supervised, active etc) are excluded. Most of the methods in the literature have tested their methods on either one or two hyperspectral datasets. This is mainly due to the lack of hyperspectral datasets with available ground truth. However, the present chapter has made an attempt to incorporate as many datasets as possible into the experimentation.

4.3.4.1 Performance Metrics Used

To quantify the performance of the proposed method, five performance metrics are used, namely, overall accuracy (OA), Cohen's kappa measure (κ), $f-score$, percentage of wrong classification (PWC) and Detection Rate (DR). The metrics are calculated from the confusion matrix given in Table 4.2.

Table 4.2: Confusion Matrix

		Actual	
		Positive Class	Negative Class
Predicted	Positive Class	True Positive (TP)	False Positive (FP)
	Negative Class	False Negative (FN)	True Negative (TN)

The overall accuracy suggests how accurately the model detects both the changed as well as unchanged pixels.

$$OA = \frac{TP + TN}{TP + FP + TN + FN}. \tag{4.4}$$

The Cohen's kappa measure takes into account actual agreement and chanced agreement. So, it is a much preferable measure for change detection purposes as the number of changed pixels is often less than the number of unchanged pixels. It is given by,

$$\kappa = \frac{OA - CA}{1 - CA}, \tag{4.5}$$

where, OA is the overall accuracy or the actual agreement while CA is the accuracy by chance expressed as

$$CA = CA_{ClassU} + CA_{ClassC}. \tag{4.6}$$

Here, the accuracy obtained by chance for unchanged class (CA_{ClassU}) and changed class

(CA_{ClassC}) are given by

$$CA_{ClassU} = \frac{(TP+FP)\times(TP+FN)}{(TP+TN+FP+FN)^2}, \tag{4.7}$$

and

$$CA_{ClassC} = \frac{(TN+FN)\times(TN+FP)}{(TP+TN+FP+FN)^2}. \tag{4.8}$$

The $f-score$ is given by,

$$f-score = \frac{2\times precision \times recall}{precision + recall} \tag{4.9}$$

where,

$$precision = \frac{TP}{TP+FP}, \tag{4.10}$$

and

$$recall = \frac{TP}{TP+FN}. \tag{4.11}$$

The percentage of wrong classification is given by,

$$PWC = \frac{100\times(FP+FN)}{TP+FP+TN+FN}\%. \tag{4.12}$$

A lower value of PWC means that the model is better.

FNR (false negative rate) suggests the rate as to how often the method predicts a pixel as changed when actually it is not. It is given by

$$FNR = \frac{FN}{FN+TN}. \tag{4.13}$$

TNR (true negative rate) suggests the rate as to how often the method predicts a pixel as a change when actually it is a change. It can be represented as

$$TNR = \frac{TN}{FP + TN}. \tag{4.14}$$

Any good change detection method should be able to give high TNR and low FNR. So, we can say, the detection rate (DR) [17], where

$$DR = (1 - FNR) \times TNR, \tag{4.15}$$

is high for a better change detection method.

4.3.4.2 Comparison with existing feature extracting models

For comparison, Window-based PCA+kmeans [132], IRMAD [133], SFI-IRMAD [134], SFI-DSP [58] and DSFANet [126] are used in the present study. S3DCAE [56] is used for comparison as the deep features extracted by it could be used for unsupervised change detection too. All these methods have used K-means as the segmentation or clustering algorithm for differentiating the changed regions from the unchanged regions. So, the proposed method has also been analysed with K-means as the function $g()$. The difference operator $\ominus$ has been chosen to be the absolute difference (AD) between the feature maps and spectral angle mapper (SAM) [135] as the comparison methods have used either one of the two.

The results for the image pair datasets are given in Tables 4.3-4.6. The results marked in bold represent the best performance and those marked in red represent the second best performance on the particular dataset. Although, it may seem a bit misleading to use accuracy for an unsupervised method, but since we already have the ground truth to compare with, we can actually see how accurately our proposed method can identify the actual changes. The detected change map for the various models are reported in Figures 4.9-4.12.

Table 4.3: Results for China Dataset

	OA	κ	$f - score$	PWC	DR
Windows PCA	0.8688	0.6801	0.8429	13.1241	0.7673
IRMAD	0.8659	0.6801	0.8408	13.4150	0.7864
SFI IRMAD	0.8784	0.7165	0.8583	12.1582	0.8284
SFI-DSP	0.8817	0.7168	0.8595	11.8316	0.8020
S3DCAE-AD	0.8885	0.7333	0.8677	11.1463	0.8108
Deep SFA	0.8920	0.7419	0.8719	10.8010	0.8166
FFCAE-SAM	0.8934	0.7489	0.8746	10.6650	0.8360
FFCAE-AD	**0.8956**	**0.7604**	**0.8804**	**10.4422**	**0.8725**

Table 4.4: Results for USA Dataset

	OA	κ	$f - score$	PWC	DR
Windows PCA	0.9106	0.7111	0.8662	8.9435	0.8182
IRMAD	0.9217	0.7471	0.8847	7.8298	0.8303
SFI IRMAD	0.9230	0.7518	0.8868	7.6959	0.8325
SFI-DSP	0.9228	0.7508	0.8863	7.7230	0.8319
S3DCAE-AD	0.9229	0.7511	0.8865	7.7135	0.8321
Deep SFA	0.9317	0.7867	0.8990	6.8323	0.8592
FFCAE-SAM	**0.9462**	**0.8421**	**0.9215**	**5.3780**	**0.9143**
FFCAE-AD	0.9447	0.8360	0.9189	5.5321	0.9057

Table 4.5: Results for River Dataset

	OA	κ	$f - score$	PWC	DR
Windows PCA	0.8863	0.4879	0.7630	11.3736	0.9562
IRMAD	0.9401	0.5663	0.7888	5.9920	0.9116
SFI IRMAD	0.9415	0.5311	0.7846	5.8530	0.8984
SFI-DSP	0.9491	0.6464	0.8261	5.0886	0.9274
S3DCAE-AD	0.9420	0.5535	0.7896	5.7966	0.9044
Deep SFA	0.9461	0.6645	0.8323	5.3933	0.9443
FFCAE-SAM	0.9584	**0.7454**	**0.8730**	4.1646	**0.9610**
FFCAE-AD	**0.9604**	0.7436	0.8720	**3.9630**	0.9519

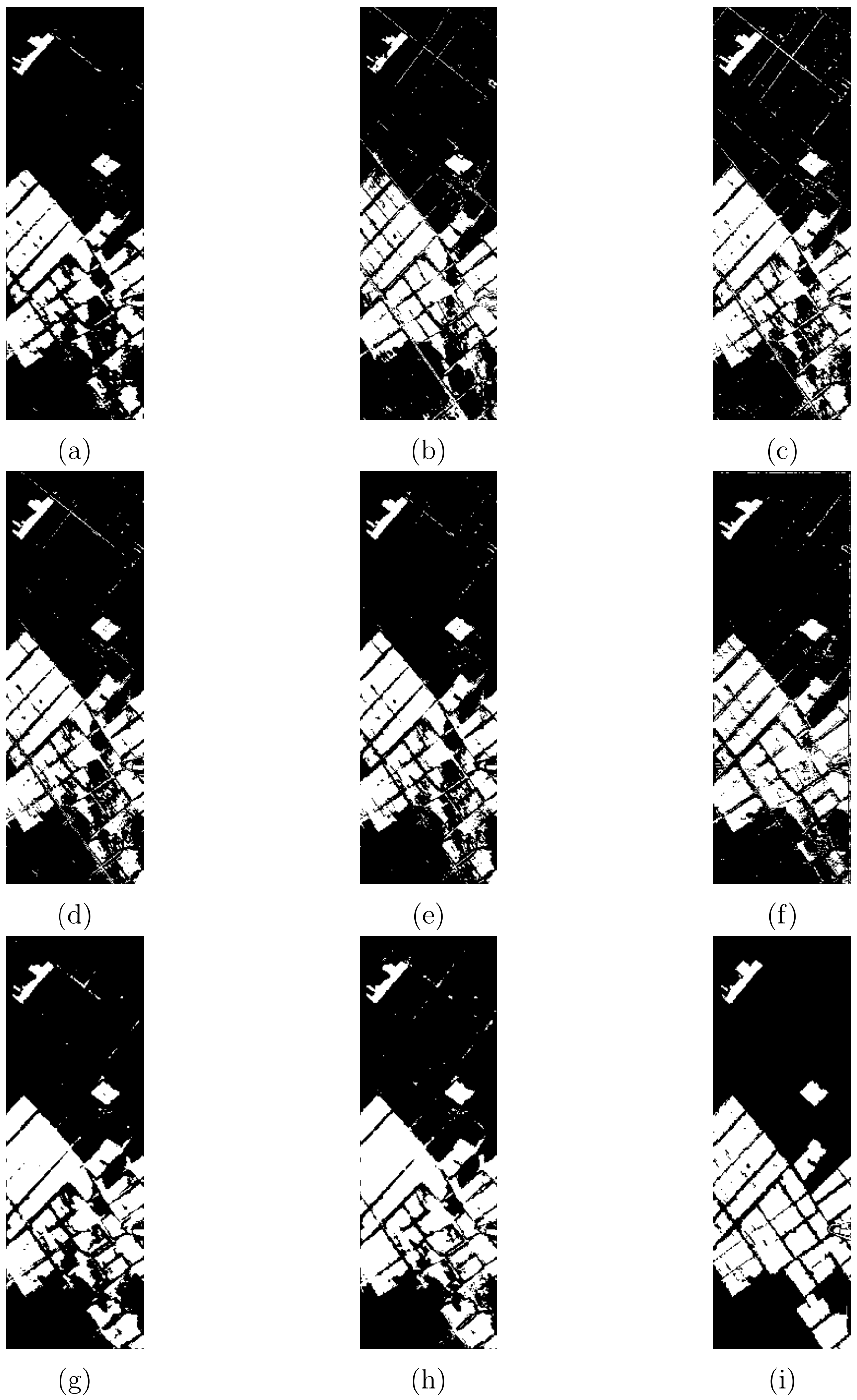


Figure 4.9: Comparison of results for China Dataset: (a) Windows PCA, (b) IRMAD, (c) SFI-IRMAD, (d) SFI-DSP, (e) S3DCAE-AD, (f) Deep SFA, (g) FFCAE-SAM, (h) FFCAE-AD, (i) Ground Truth

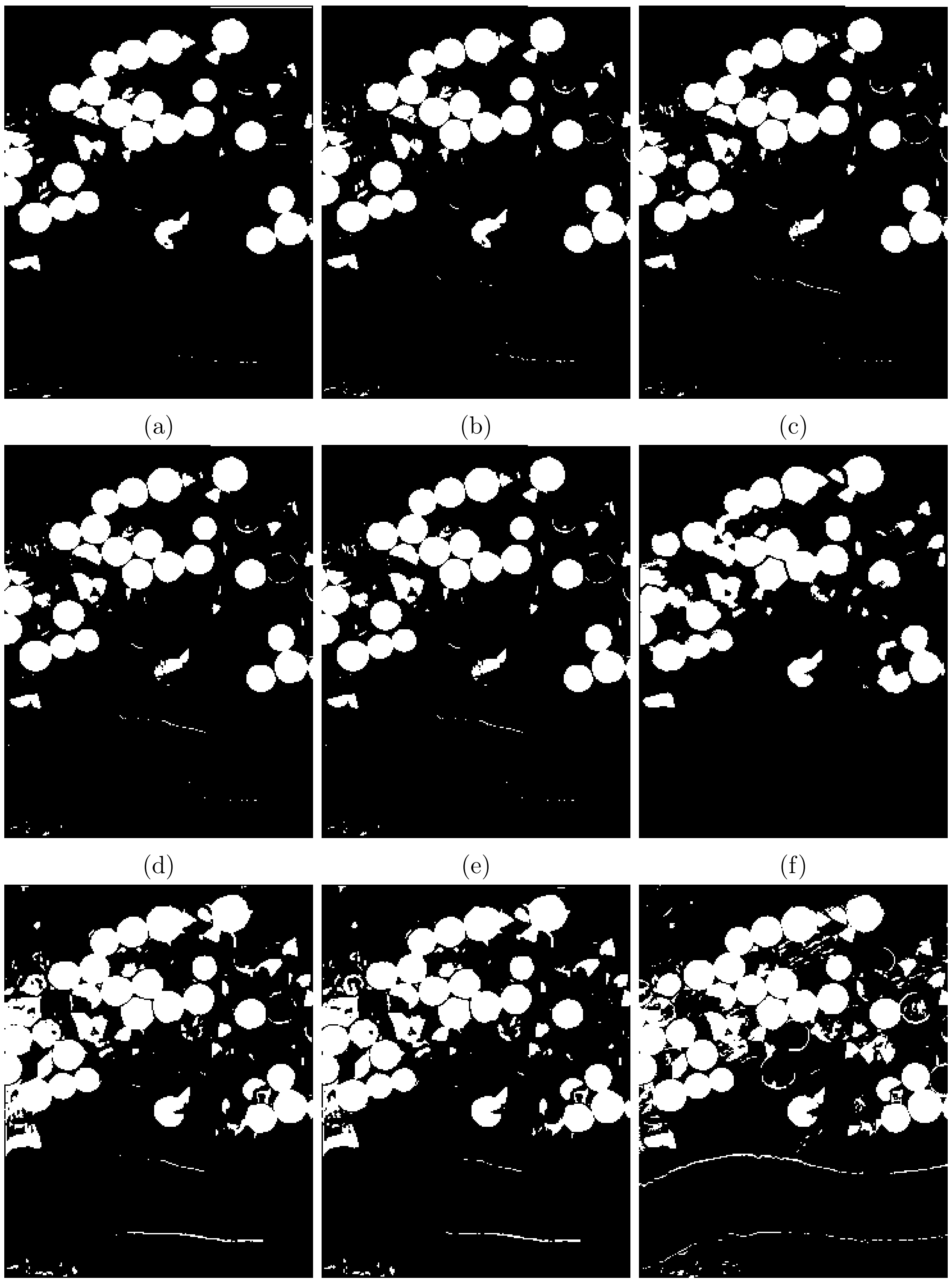


Figure 4.10: Comparison of results for USA Dataset: (a) Windows PCA, (b) IRMAD, (c) SFI-IRMAD, (d) SFI-DSP, (e) S3DCAE-AD, (f) Deep SFA, (g) FFCAE-SAM, (h) FFCAE-AD, (i) Ground Truth

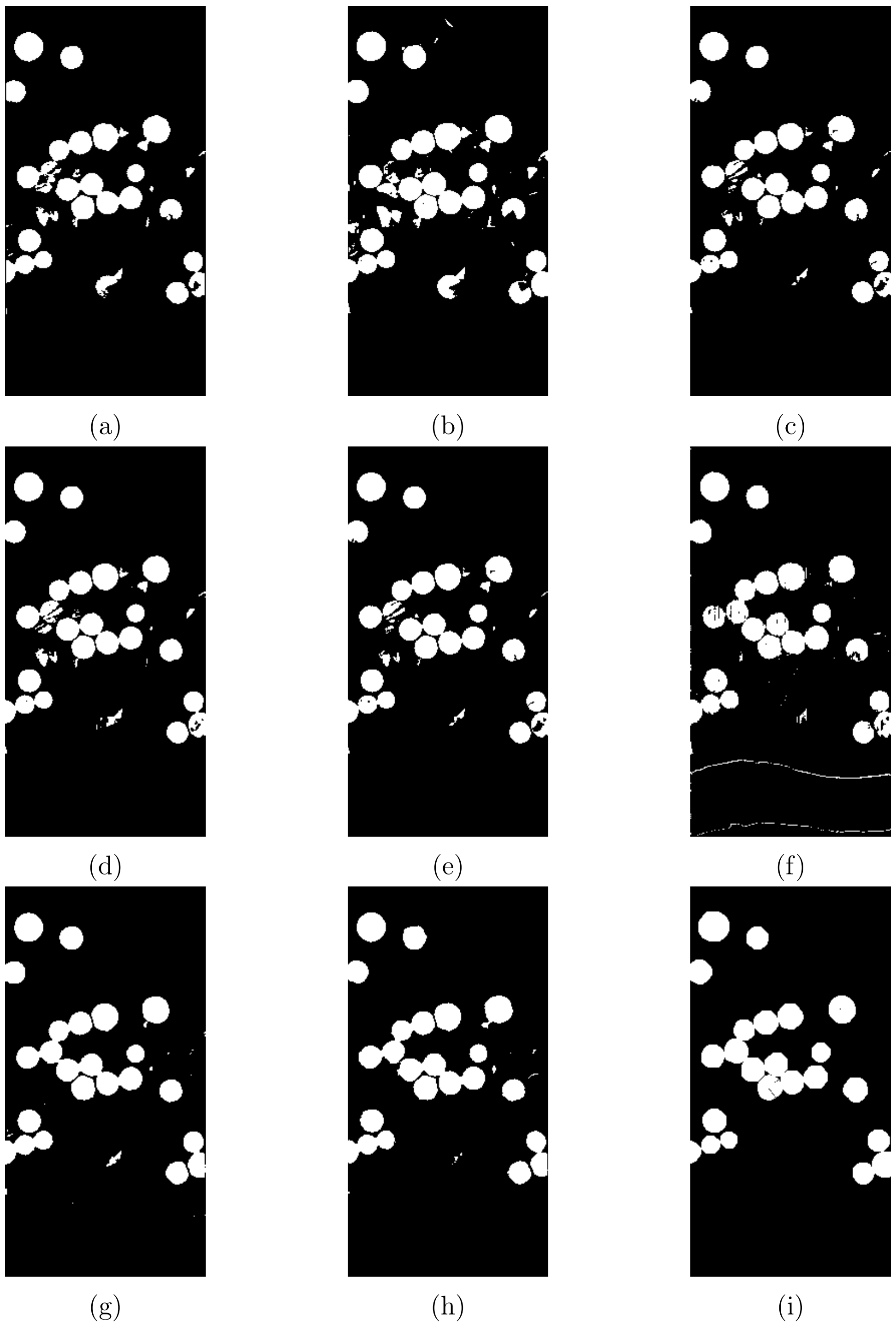


Figure 4.11: Comparison of results for River Dataset: (a) Windows PCA, (b) IRMAD, (c) SFI-IRMAD, (d) SFI-DSP, (e) S3DCAE-AD, (f) Deep SFA, (g) FFCAE-SAM, (h) FFCAE-AD, (i) Ground Truth

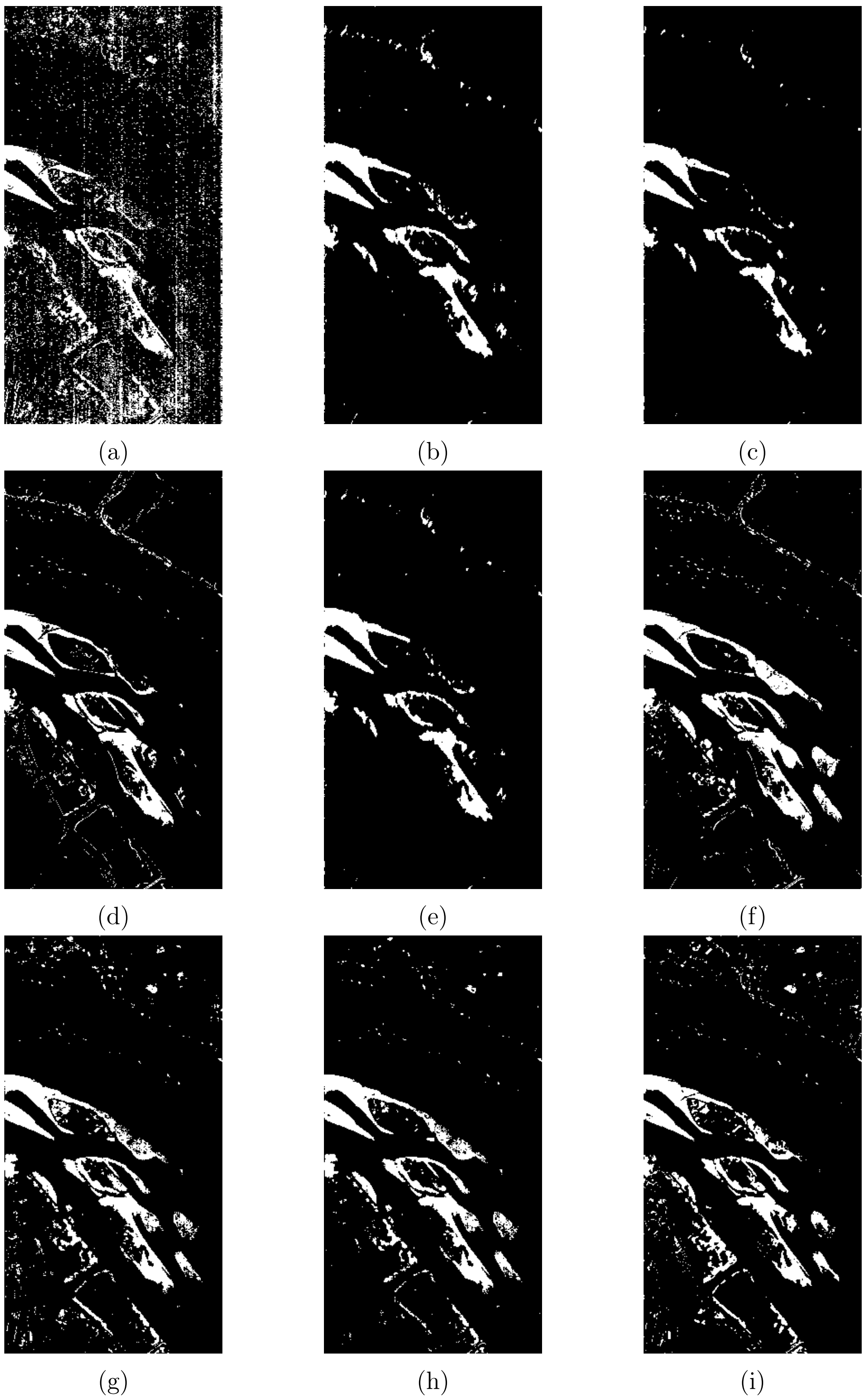


Figure 4.12: Comparison of results for Hermiston Dataset: ((a) Windows PCA, (b) IRMAD, (c) SFI-IRMAD, (d) SFI-DSP, (e) S3DCAE-AD, (f) Deep SFA, (g) FFCAE-SAM, (h) FFCAE-AD, (i) Ground Truth

Table 4.6: Results for Hermiston Dataset

	OA	κ	$f-score$	PWC	DR
Windows PCA	0.9670	0.8503	0.9252	3.3013	0.9584
IRMAD	0.9618	0.8348	0.9178	3.8154	0.9679
SFI IRMAD	0.9751	0.8837	0.9426	2.4936	0.9588
SFI-DSP	0.9752	0.8872	0.9437	2.4795	0.9666
S3DCAE-AD	0.9751	0.8842	0.9427	2.4872	0.9594
Deep SFA	0.9742	0.8833	0.9417	2.5795	0.9674
FFCAE-SAM	0.9812	0.9145	0.9574	1.8756	0.9727
FFCAE-AD	**0.9819**	**0.9176**	**0.9589**	**1.8051**	**0.9730**

4.3.5 Statistical-significance test

The performance of different models is assessed by several performance scores. We should not, however, accept any of the metrics in isolation to prove that one is better than the other. No metric alone can encompass every area of interest. Even for one element of interest, several metrics must be given to detail the performance of a model. The statistical tests often evaluate the superiority of the a model by encompassing a single metric only.

Therefore, this chapter shows a ranked ordering of the models taking all the performance metrics into account. The average ranking of the compared models on all the four datasets on the five metrics are reported in the Table 4.7. To avoid rewriting the names of the models, the models are denoted as A (Windows PCA), B (IRMAD), C (SFI-IRMAD), D (SFI-DSP), E (S3DCAE-AD), F (Deep SFA), G (FFCAE-SAM) and H (FFCAE-AD).

It can be seen that the proposed FFCAE with SAM and AD have more or less similar ranks, but are better in ranks than the others. For statistical test, one can perform a Tukey HSD test [139] as a post-hoc test to justify the superiority of the proposed model. There are eight models ($k = 8$) being compared against each other and thus 28 pairs of

Table 4.7: Average ranking of the compared models on all the four datasets on the five metrics.

	A	B	C	D	E	F	G	H
Accuracy	7.50	7.50	5.25	4.25	4.50	4.00	1.75	1.25
Kappa	7.75	6.75	5.50	4.50	4.75	3.75	1.50	1.50
f-score	7.50	7.25	5.50	4.50	4.50	3.75	1.50	1.50
PWC	7.50	7.50	5.25	4.25	4.50	4.00	1.75	1.25
DR	6.50	5.75	5.50	5.50	5.75	3.75	1.50	1.75

Table 4.8: Value of Tukey HSD Q-statistic for the models compared.

	A	B	C	D	E	F	G	H
A	0	2.11	**10.26**	**14.47**	**13.42**	**18.43**	**30.27**	**31.06**
B		0	**8.16**	**12.37**	**11.32**	**16.32**	**28.16**	**28.95**
C			0	4.21	3.16	**8.16**	**20.00**	**20.79**
D				0	1.05	3.95	**15.79**	**16.58**
E					0	**5.00**	**16.84**	**17.63**
F						0	**11.84**	**12.63**
G							0	0.79
H								0

models to compare. Degrees of freedom for the error term is $\nu = k(k+1) = 32$. The error obtained is 5.775 and since the degree of freedom for error $\nu = 32$, the mean-square error, $MSE = 5.775/\nu = 0.1805$. Thus, the Tukey HSD Q-statistic is given as,

$$Q_{i,j} = |\bar{r}_i - \bar{r}_j|.\sqrt{\frac{n}{MSE}}. \tag{4.16}$$

Here, $\bar{r}_i$ and $\bar{r}_j$ correspond to the performance of the i^{th} and j^{th} model respectively. The critical value at 95% confidence ($\alpha = 0.05$ significance level) $Q_{critical}^{\alpha=0.05,k=8,\nu=32} = 4.5209$. Table 4.8 shows the Tukey HSD Q-statistic for the compared models. Any value greater than $Q_{critical}^{\alpha=0.05,k=8,\nu=32}$ implies significance. The values marked in bold suggest that the difference is significant and those marked in red suggest insignificance. It is evident that the proposed models FFCAE-SAM and FFCAE-AD (renamed as G and H, respectively for simplicity) are significantly better than all other models. However, there is no significant difference between FFCAE-SAM and FFCAE-AD which is quite intuitive as the proposed feature extraction method is the same for both.

4.4 Deep Convolutional Autoencoder for Partially Labelled Hyperspectral Images

The classification performance is often influenced by the quality of labelled samples used during the training. Manual generation of training set is often accomplished by visual assessment of the scene and sequential labelling of each pixel. This step is both time intensive and very redundant. In reality, the training set includes numerous neighborhood pixels with the same information. Although such redundancy is not hazardous to the quality of the outcomes if handled appropriately, it significantly slows down the training process. To make the models as efficient as feasible, the training set should be kept as small as possible and concentrated on the pixels that truly contribute to increase the model's performance. Algorithms involving active learning [140, 141], on the other hand, can assist in overcoming the paucity of training data.

This section presents an active learning for semi-automatic change detection in hyperspectral images. A convolutional autoencoder (CAE) [96] based model is used which would reduce the dimensionality of the pixels. The autoencoder based model is particularly suited in this context as it can learn features from unlabelled data as well. In that case, the re-training with labelled samples would help in fine-tuning. The proposed classifier uses a twin encoder and scores the unlabeled pixels. At each epoch it automatically selects top k samples deemed most uncertain, thus reducing the need for a large training set. The label information of the selected pixels are provided to the model for improvement, and the procedure is iterated by adding the labelled samples to the training set. Starting with a short and non-optimal training set, the model constructs the ideal collection of samples to minimise classification error. The outcomes of the experiments confirm the consistency of the procedures.

4.4.1 Related Research

Active learning algorithms [140, 141] actively seek labels from the user or oracle. At each iteration the model itself chooses a few uncertain samples and queries the oracle about the label information. Because the model selects the instances, the number of examples necessary to train the model is substantially smaller than the number required in traditional supervised learning. Active learning algorithms are widely used in environmental monitoring [142] and classification [143], species classification [144], multi-label classification [145], classification of hyperspectral images [146], etc. Recently, active learning has been adopted in the change detection literature [62]. The complexity of labelling each pixel (as changed or unchanged) in a pair of hyperspectral images, thus makes it particular match for active learning application, which requires comparatively lesser labelled samples. The high dimensionality of these images, on the other hand, is another interesting property which is both useful as well as a challenge.

Deep learning [8, 9] has emerged as a relatively impressive technique for dimensionality reduction. In recent years, they have been notably used for analysing remotely sensed satellite images [147] and change detection [141]. Deep learning has been paired with active learning for change detection in areal images [148]. However, the literature on change detection in hyperspectral images with active learning is still lacking. This is mainly because of lack of labelled data as well as high dimensionality. In these cases, unlabelled data are abundant. Thus, autoencoder (AE) based feature extraction is an inevitable choice [8, 9]. An AE can be used to learn repeated patterns [16] because it maps similar values to similar ranges. So, pixels which are similar in nature will be efficiently mapped to similar range in the reduced dimension too. Borrowing this idea, this article proposes a CAE based model which would reduce the dimensionality of the pixels. The model is then fine-tuned by stacking the encoder of the CAE on a classification layer which is trained using active learning.

4.4.2 Proposed Method

The proposed method utilises active learning to train a CAE pre-trained convolutional neural network which would classify the changed pixels from the unchanged pixels in a semi-automated manner. The subsequent section describes the dataset on which the experiments have been performed.

4.4.2.1 Dataset Description

In this study, two pairs of hyperspectral images are considered for analysis. These two datasets have incomplete ground truth data i.e. they have some unknown pixels along with the labelled change and unchanged pixels. The first dataset is Bay Area [149] which was taken on the years 2013 and 2015 respectively with the Aviris sensor surrounding the city of Patterson (California). There are 224 spectral bands and the image size is 600 $\times$ 500. In the ground truth data, there are 39,270 pixels which are labelled as changed, 34,211 pixels which are labelled as unchanged and 2,26,519 pixels which are unlabelled or unknown. There is a similar scenario with the second dataset Santa Barbara [149] which was taken on the years 2013 and 2014 with the Aviris sensor over the Santa Barbara region (California). There are 224 spectral bands and the image size is 984 $\times$ 740. In the ground truth data (Figure 4.13), there are 52,134 pixels which are labelled as changed, 80,418 pixels which are labelled as unchanged and 5,95,608 pixels which are unlabelled or unknown.

4.4.2.2 Unsupervised pre-training of Sliding Convolutional Autoencoder

The proposed method a single CAE for feature extraction which slides over the two images. The network architecture is shown in Figure 4.14. The two networks slide across the images by a fixed stride of one. To avoid boundary issues, the images are suitably padded with zeros. The network takes a small (9 $\times$ 9) neighborhood of pixels and tries to reconstruct it back at the output. In order not to increase the computational complexity

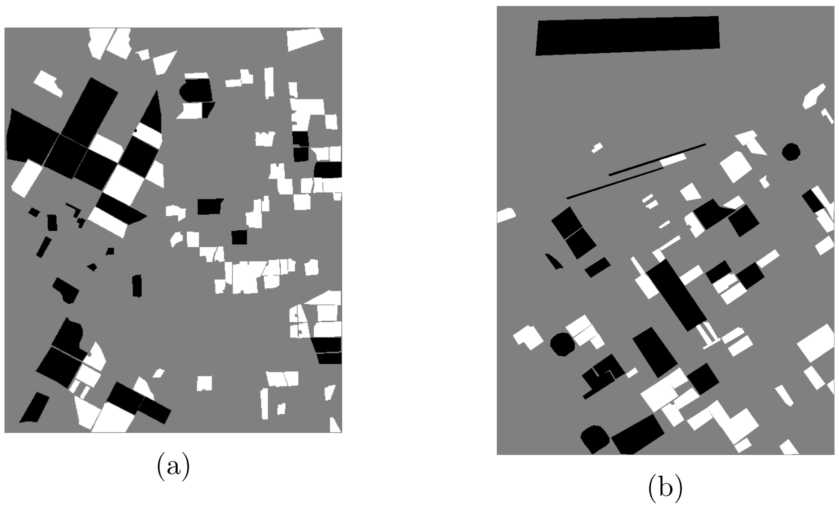


Figure 4.13: Ground Truth for (a) Bay Area and (b) Santa Barbara dataset. (Black- Unchanged, White- Changed, Gray- Unknown).

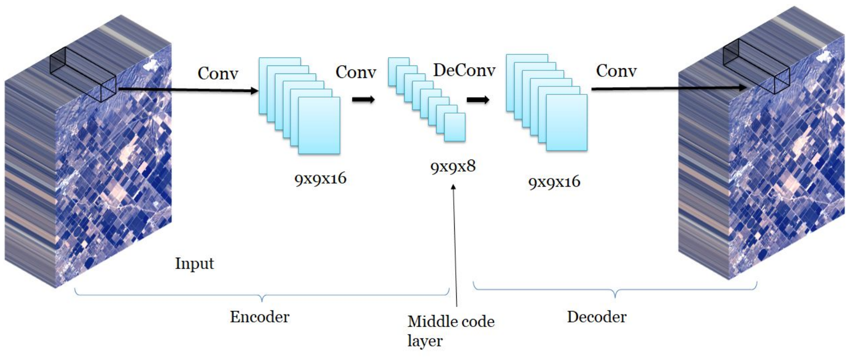


Figure 4.14: Architecture of the CAE network used.

and to avoid the effects of noisy points, the neighborhood is not taken any larger or smaller respectively. The CAE is trained for 50 epochs with adam optimiser to minimise the mean squared error of reconstruction. The input to the network is, therefore $9 \times 9 \times 224$. The network learn the spatial as well as spectral features and reduces the dimensionality of the input space while trying to reconstruct it back. Since an AE maps different input values to different ranges [16], the CAE will learn to map the unchanged pixels from both the images into the same range as both the images would have similar intensity values for those pixels. On the other hand, if the pixels represent a changed region, the intensity values will be different, and hence the CAE will map the pixel values

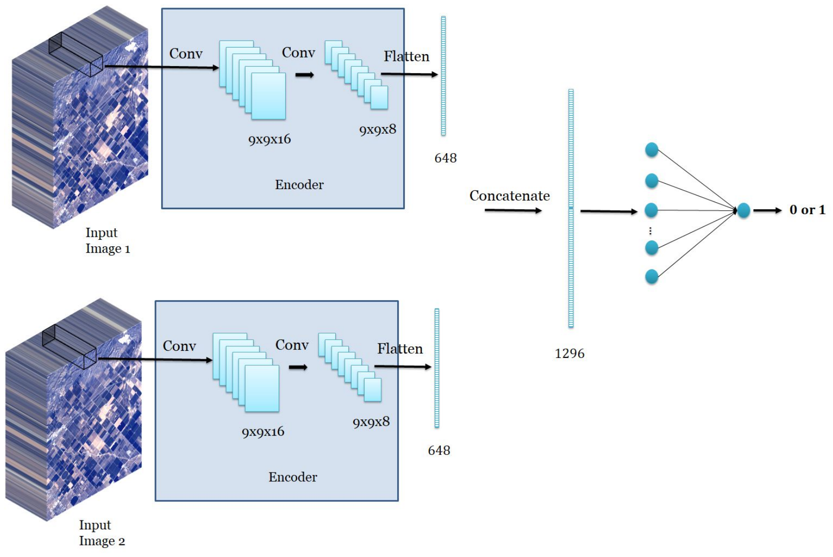


Figure 4.15: Architecture of the classifier network used.

for the two images to different range. The network will, in a way, learn to distinguish between the changed and unchanged pixels. The CAE is useful because it can also make use of samples which are unlabelled. Using active learning based re-training, the network can be fine-tuned further with very few labels.

Table 4.9: Comparison of the results obtained on the two datasets by various models.

	Bay Area			**Santa Barbara**		
	ROC-AUC	**Accuracy**	**F-Score**	**ROC-AUC**	**Accuracy**	**F-Score**
Proposed	**0.9979**	**0.9843**	**0.9843**	**0.9983**	**0.9851**	**0.9844**
Method 1	0.9915	0.9646	0.9649	0.9915	0.9694	0.9679
Method 2	0.9870	0.9559	0.9560	0.9879	0.9647	0.9630
Method 3	0.9923	0.9645	0.9649	0.9868	0.9647	0.9629

4.4.2.3 Active re-training of Change Detector Model

After the CAE is trained, the network is replicated into two identical copies (one for each image). Two images are simultaneously passed through the encoders they become input to a fully connected classification network . The label information of a pixel at the final layer corresponds to the central pixel of the 9 $\times$ 9 input neighborhood. The labelled samples in the dataset are divided into training and testing set in a 70% and 30% proportion respectively. In the first epoch, the network Figure 4.14 is provided with a random percentage (between 10-20%) of the training labels for forming the initial decision boundary. This is the initial tiny training set for the model assumed to be marked by the experts. After each epoch, the network would make predictions based on the current decision boundary and would decide on the most uncertain pixels. The uncertainty of the pixels depend on how close they are to the decision boundary which is characterised by the output (O) of 0.5 (suggesting a mid point between the classes represented by 0 and 1 at the output). The uncertainty is modelled as a beta distribution function given by,

$$U = \frac{O^{\alpha-1}(1-O)^{\beta-1}}{\mathbf{B}(\alpha,\beta)}, \tag{4.17}$$

where α and β are the shape parameters. The beta function $\mathbf{B}$ is usually represented in terms of gamma functions (Γ) as,

$$\mathbf{B}(\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}, \tag{4.18}$$

for normalisation purposes. In this case, the values of α and β are chosen to be 2. This indicates that if the output O is close to 0 or 1 suggesting a high confidence of belonging to either one of the classes, it will not be treated as an uncertain sample as the value of U would be close to zero. However, a point lying close to the decision boundary will generate an output near to the midpoint of 0.5 and the value of U would be close to 1. If the samples are ranked in decreasing order by U at the output, the most uncertain samples will be the topmost one. At each epoch the model is allowed to choose only the top k of those uncertain samples (or pixels in this case). The labels to those chosen pixels are provided to the model and those samples are included in the training set. The model is trained for 100 epochs with Adam optimiser to minimise the cross-entropy loss.

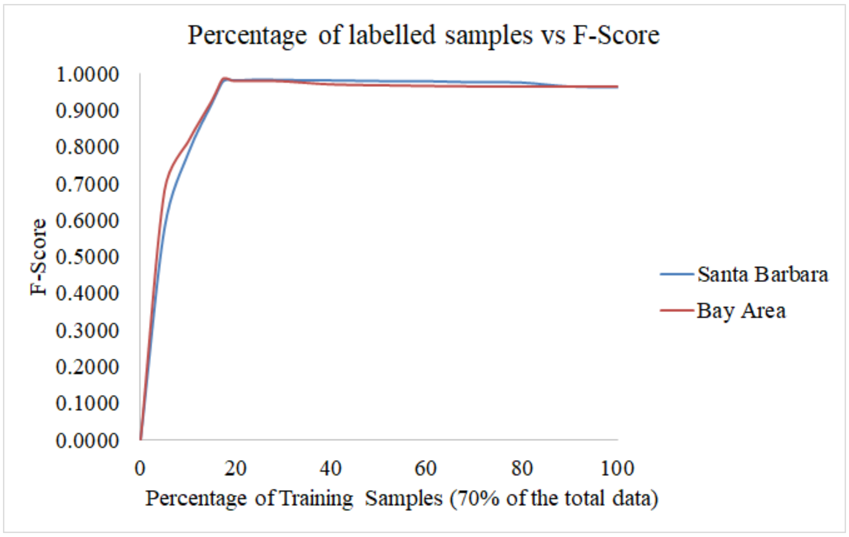


Figure 4.16: Variation of F-Score with the percentage of training samples

4.4.3 Results

The model is compared with the proposed FFCAE based unsupervised change detection (Method 1) and a semi-supervised [125] (Method 2) method. Due to the lack of change detection methods in hyperspectral images involving active learning, active learning based models could not be analysed. The model is also compared with a case where the same proposed model is trained on the full training set (70% of the labelled samples) (Method 3). This would be a fully supervised model. The results are reported on the testing set for all the models.

4.4.3.1 Variation of F-Score with the percentage of training samples

In order to decide on the value of k, we used 10-fold cross validation on the initial tiny training set provided to the model. It has been observed that the optimal value of k for the Bay Area images was 12 and for Santa Barbara images was 24. After 100 epochs, the model needed only 8817 and 17231 training samples for the Bay Area and Santa Barbara images simultaneously which constitute only 17.14% and 18.57% of the training samples. The variation of F-Score with the percentage of training samples is provided in Figure 4.16.

4.4.3.2 Comparison with other methods

Pre-training the network using the sliding CAE resulted in a form of self-supervised feature extraction. Re-training with labels just fine-tuned the extracted features for the desired classification task. Since such semi-supervised models are scarce, the sliding CAE based model was compared with the unsupervised FFCAE method proposed in the section 4.3.2. The change maps generated by the various models are provided in Figures 4.17 and 4.18. For most of the pixels, the labels are unknown. Hence the change map may be used as a supplementary specimen for the human annotator.

It would have been unfair to compare the proposed sliding CAE with other existing semi-supervised or active learning based models which are not built for HSIs. Almost all such models handle only a few selected bands from SAR images or multispectral images. So, the capabilities of sliding CAE as a feature extractor can only be argued on the basis of its light-weight implementation and suitability for bigger sized images. The results of comparison of performances for different methods are reported in Table 4.9.

4.5 Conclusion and Future Work

One of the major challenges faced in these hyperspectral change detection problems are the under-detection and over-detection of changed pixels. They are usually caused due to similarity in the intensity values of the changed and unchanged regions, faulty classification by colour thresholding or texture features. If there are many types of changes occurring over long periods of time, then this makes the task much more complicated. The changes are usually distinguished by their characteristic spectral signatures and thus, some minute changes might often get overshadowed by a prominent large change. The proposed FFCAE has been aimed to handle these issues. The results obtained for the four hyperspectral datasets show promising feature extraction capabilities of the FFCAE.

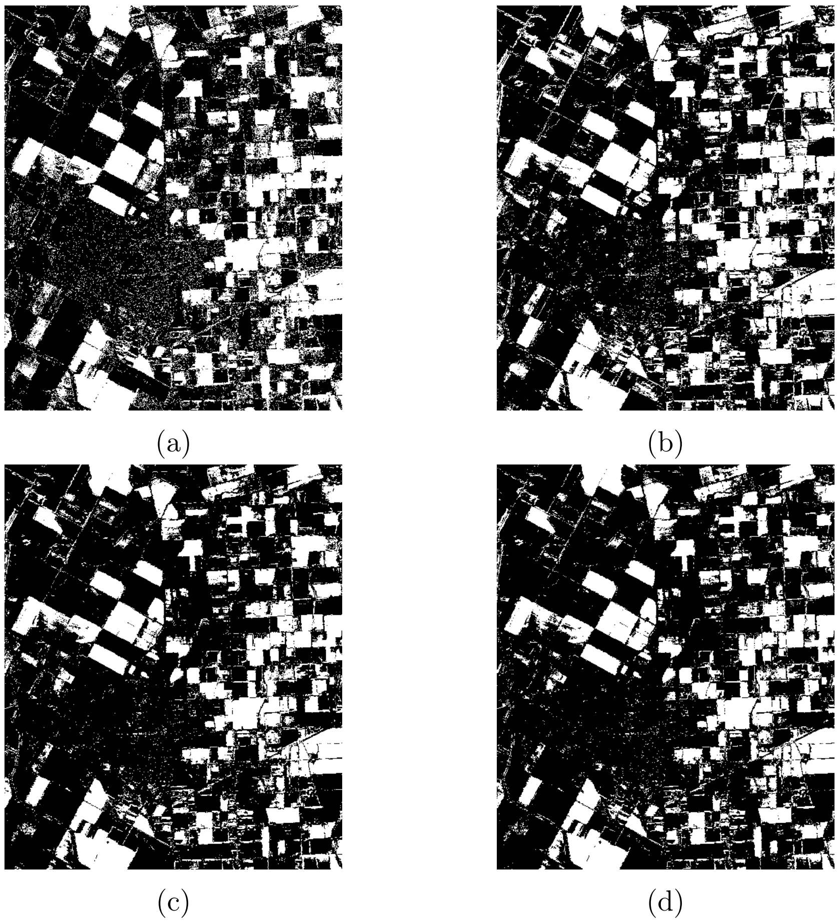


Figure 4.17: Change map (Black- Unchanged, White- Changed) obtained for Bay Area dataset by the (a) proposed model, (b) Method 1, (c) Method 2 and (d) Method 3.

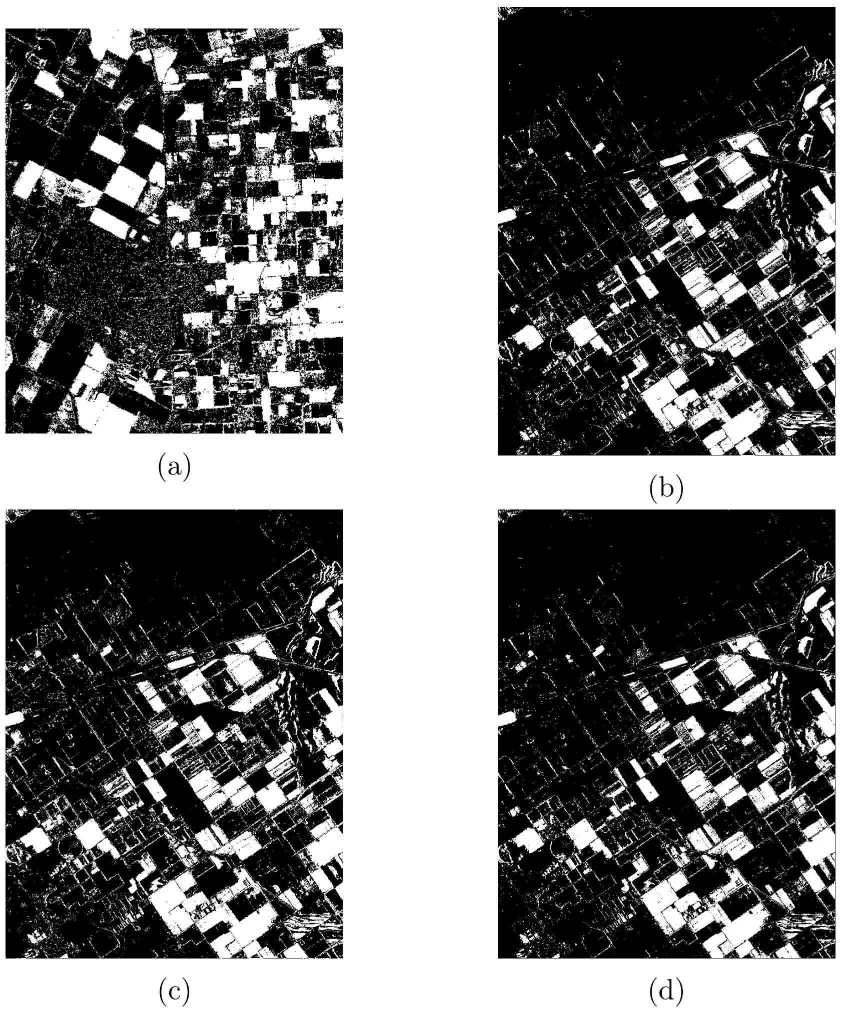


Figure 4.18: Change map (Black- Unchanged, White- Changed) obtained for Santa Barbara dataset by the (a) proposed model, (b) Method 1, (c) Method 2 and (d) Method 3.

We can safely claim that the proposed method of feature extraction performed better than all the methods compared with respect to the four datasets and all the five metrics. The proposed method takes only twin filters into account. Increasing the number of filters would give rise to situations of overfitting and using filters of size more than 5×5 would only contribute to the effect of suppressing the small changes. However, the consequence of increasing the depth in detecting changes is a matter of further investigation and needs a separate study in itself. In that case, the presence or absence of skip connections may be further investigated from the direction of residual networks. However, it was clear that an autoencoder based training (as done for FFCAE) is significantly better than the state-of-the-art feature extraction models for unsupervised change detection in a pair of HSIs.

However, if limited labelled data is present, then unsupervised FFCAE based model might give poor results compared to models which make use of that limited amount of labels. As observed from the experiments, the active learning based model used very little (only 17.14% and 18.57%) training data to achieve an elevated accuracy. Surprisingly, the proposed active learning based model could achieve better accuracy than the corresponding supervised model (which used 100% of the training data) even after using a comparatively lesser amount of training data. This is because active learning uses a concept of smart selection of labelled samples rather than the random selection as in supervised learning. Since the datasets are incompletely labelled, the proposed model can also be used as a complementary option for finding the labels of the pixels. However, it is observed [Figure 4.17] that many tiny localised regions of changes are predicted by the model for the unknown pixels. The information about these pixels being actual changes or simply noise can only be judged after they are validated by a domain expert.

Although this chapter has compared the suitability of CAE in supervised and semi-supervised change detection, the capabilities or adaptations of FFCAE feature extraction in a supervised or semi-supervised setting and that of sliding CAE in unsupervised setting is still left unexplored. Moreover, the works described in this chapter concern only with binary change detection where the problem only is to identify between changed and unchanged pixels. Including the label information in the dimensionality reduction process may also be extended to multi-class change detection as well. Research should also be extended to cases where a pixel changes from one class to other over time. This can also be explored under the dynamic data setting to study evolving classes due to climate changes.

Chapter 5

Autoencoder based Hybrid Multi-Task Predictor Network for Daily Stock Prices Prediction

5.1 Introduction

In order to study the dexterity of AEs in chaotic time-varying datasets, this chapter presents a stock price prediction system. Stock price trends [150] is inherently very difficult to model due to its uncertain nature but has become one of the widely researched topics among the machine learning enthusiasts. Investors look forward to statisticians to predict the stock market trends. However, using traditional statistics, the prediction of prices becomes one of the most difficult tasks to do because of its dependency on multiple issues like physical factors, foreign markets trends, domestic market trends, news, people sentiments etc. This task is a multivariate time-series prediction problem. Such data is highly chaotic and can be treated as a complex adaptive system or dynamic system [151]. The problem is often tackled in two approaches. Researchers either look at market factors and try to predict the movement of stocks or study the historical to predict the future prices.

Thus, stock market analysis is broadly of two types: fundamental analysis [152, 153] and technical analysis [154]. The first one concentrates on the current economic and financial situation of the company, considering the company's assets, market value, debt inquired, profit, loss, balance sheet and other major economic reports describing the performance of the company. The second one studies the past market actions and price values, to determine its future performance. To develop machine learning models, the primary focus is usually on the price of the stocks, and the volume of stocks that are traded on a regular basis, and other factors include the supply and demand that captures the movement of the stock market.

In his theory, Burton [155] points out that the stock market forecasting or modeling is not practical, as price fluctuations are unexpected in the actual world. Every movement in financial market pricing is dependent on current economic events or announcements. Traders are profit-oriented, and their judgments are taken based on most current occurrences, irrespective of prior analyses or expectations. The dispute about this hypothesis of an efficient market has never been resolved. There is no substantial evidence to show whether or not the efficient marketing theory is correct. But stock markets, according to Y. S. Abu-Mostafa [156], are to some extent predictable. The future finance markets are impacted by previous experience of numerous price fluctuations over a period in the stock sector and from the unreported serial connections of critical economic situations.

Stock prices from history are used to predict the future value of a stock corresponding to the stock exchange. These predicted stock prices helps the investors or traders to decide whether they want to trade stocks now or hold them for some time. Therefore, stock market analysis can be broadly classified into, short term trading, high-frequency trading, long-term trading, intraday trading [157] and interday trading [158]. This work concentrates on buy-today and sell-tomorrow (BTST) trading only. Traders buy and sell stocks in large numbers with the intention of booking the highest amount of profit in a day. Such trading, being short term, is a lot more riskier than investing in the regular stock market. So, it is very crucial to know the prices of the stocks on the next day to gain profit. Various models [159, 160, 161] have been proposed in this regard to predict the prices of stocks. However, almost all the models try to predict the movement of the prices using some indicator variables [162] which are mostly modelled on the opening, high, lowest and closing (OHLC) prices of a stock on a particular day. Studies on direct prediction of OHLC prices are not so widespread in the machine learning community. Although there have been some attempts to predict the closing prices [163], high prices [164] or the average prices [165] independently, the literature has somewhat ignored the possibility of modelling OHLC prices together.

With the advancement in the field of neural networks, it has become comparatively easier to model the stock prices variations up to a fairly good extent [63]. Long Short Term Memory (LSTM) networks [166] and Gated Recurrent Unit (GRU) networks [167] have become the state of the art models for efficiently predicting the stocks. But, these neural networks come with a cost. If the normalisation of the input variables are not done carefully, the predictions often end up being faulty [168]. This calls for a new variation in normalisation of stock prices. Moreover, since there is an angle of financial risk to these models, they need to be as accurate as possible in their predictive capabilities. As argued by most deep learning researchers [169], recurrent neural networks (RNNs) perform better predictions if the raw data is transformed to a different feature space. There have been studies [17] which show that autoencoders are good feature extractors. This has been the motivation consider such autoencoder based feature extracting methods for predicting stocks. Combined with the time series forecasting abilities of LSTM networks and feature learning capabilities of AEs, this chapter presents a hybrid network which takes help of multi-task learning to learn the OHLC relationships. It is seen that using the proposed model, the most profitable stocks can be recommended efficiently. A momentum indicator known as William's %R [154] is used to determine the stocks which would be overbought the next day. Such a recommender system would help the investors make profitable decisions while trading.

Existing research on predicting open, high, low and close price (OHLC) data suffers from two flaws. There are two main reasons behind this. This means that all values must be positive, the low price must always be less than or equal to the high price, and that both the open and close prices must fall between two defined limits in order for the OHLC data to be valid. Constraints must be taken into account when using the time-series modelling techniques for the four variables of OHLC data. The contributions of this chapter is three fold, namely,

(i) A novel normalisation approach for the prediction of OHLC prices,

(ii) a novel hybrid multi-task network of AE and LSTM-RNN, and

(iii) a system to recommend the most profitable and overbought stocks (using William's %R indicators) for the next day.

The remaining chapter gives a detailed overview of the work presented. The following section 5.2 contains the description of related works. Section 5.3 gives a preliminary description of the building blocks of the proposed method. Section 5.4 contains the dataset description, data pre-processing, and describes the working of the system. In section 5.5, experimental results are shown containing all the result visualization. The conclusions drawn from the proposed system has been provided at the end in section 5.6 followed by the scope of future works.

5.2 Related Works

A commendable amount of work has been done in the field of stock price prediction. Past studies [170] have shown that it is impossible to predict the stock prices accurately as it depends on so many factors including the domestic market trends, population sentiments, news, foreign market trend, etc. The stock price time series follows a random walk model which makes it less likely to be predictable. However, with the recent advancements in the field of machine learning, it has become possible to predict the stock prices accurately up to a fairly good extent.

There has been a benchmark survey [159] of techniques including Logistic Regression, Support Vector Machines (SVM), Artificial Neural Networks (ANN), k-Nearest Neighbor (kNN) and other ensemble models, which can predict stock prices. Researchers [171] implemented SVM along with kNN to analyse stock market trends on Indian stocks. Researchers [172, 173] have been much appreciative of autoregressive models like ARIMA (auto-regressive integrated moving average) which have been proven to be robust for short-term forecasting. A triple stage method was proposed [174] for forecasting the stock prices. It included a stepwise regression analysis, grouping of selected variables by a differential evolution-based fuzzy clustering method and a fuzzy inference neural network for final prediction. A similar work [175] has also been proposed which combines dimensionality reduction using principal component analysis (PCA) with ANN for S&P 500 Index exchange-traded fund (SPY) stock market returns. Hybrid models of fuzzy time series and automata [176], granular computing [177] were also explored. Support vector regression (SVR) models have been explored [178] for stock price prediction. Hybrid of neural networks [179] have also been used to predict the volatility of stocks.

Deep neural networks are the state of the art models [160] for stock prediction. Recently, research has been diverted to using LSTM [180, 166, 181] models. Researchers [161] have used LSTM and GARCH [182] to forecast the volatility of stock price index. A standard LSTM network [180, 181] or a standard GRU network [183] would simply takes in the stock prices and predicts the next day's price. LSTM networks have been used to predict the closing price of the stocks [163] recently. However, there are only a handful of works to the knowledge of the authors which have handled the prediction of

OHLC prices. It may be argued that predicting OHLC prices directly makes more sense as the technical indicators used for stock price modelling are built on these OHLC values only and a novice trader will not understand the meaning of these indicators. However, anyone can understand that one can make high profit is he buys a stock closer to the low price and sells it on the highest price. The bullish and bearish signals can also be judged by looking at the opening and closing price for the next day. If the opening price is greater than the closing price, it is a bearish signal and indicates that the market is going to be unfavourable. On the other hand if the opening price is lesser than the closing price, it is a bullish signal and indicates that the market is going to be favourable. Thus, the present study concentrates on efficient prediction of OHLC prices directly.

A novel loss function based LSTM network has been proposed to predict the opening, high, lowest and closing prices separately known as FLF-LSTM [184]. Even though they have not explicitly identified or handled the OHLC constraints, it is found that the model is capable of making meaningful OHLC predictions. Another research [165] proposed predicting the average of the OHLC using an LSTM network, but was eventually unable to make meaningful predictions as seen in the experiments. A bi-directional LSTM based multi-task model has been proposed to predict OHLC prices [185]. It is shown [186] that multi-task models which make use of hard parameter sharing [187] are less prone to overfit and can efficiently learn the relationships in the data. However, the proposed model fails to make any meaningful predictions and the predicted values do not follow the constraints of an OHLC data (explained in section 5.3.2). This is mainly because their model assumes independence of the OHLC values in their proposed model as they start with the task-specific layers in the initial layers and then try to aggregate using a shared layer in the latter part of the network. Thus, the proposed work aims to take a critical angle at predicting meaningful OHLC prices and also reducing the financial losses incurred by following the less accurate stock price prediction models. The following section explains some preliminary concepts used in the proposed method.

5.3 Preliminaries

Stock market predictions are used to estimate the true value of stock for the upcoming days. Basically, the aim is to understand the future behaviour of the stock market on the basis of previous performance of the stocks and the information about the company's stocks.

To analyze the stocks is a very challenging task, as it requires a deep understanding of the stocks and market trends, and of other factors, including current industrialized policies, effect of foreign markets, country's financial situation, past market behavior, etc. The other major difficulty is the volatility of the stock market [150, 188] which makes it nearly impossible to understand the trends in the stock price. With the increasing research in the field of RNNs, such time varying trends are easily recognised. The following section gives a very brief overview of the LSTM cell and the OHLC properties.

5.3.1 Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM) [189] cell is a special kind of RNN node with a forget gate. A LSTM cell consist of four parts i.e. input gate ($\mathbf{i}_t$), output gate ($\mathbf{o}_t$), cell state ($\bar{\mathbf{C}}_t$), and forget gate ($\mathbf{f}_t$). The LSTM cell[166] takes three inputs, the output from previous unit $\mathbf{h}_{t-1}$, the memory of the previous unit (cell state) $\mathbf{C}_{t-1}$, and the input of the current time step $\mathbf{x}_t$.

This single unit makes decision by considering the three features and produces a new output, and alter its memory. $\mathbf{C}_t$ and $\mathbf{C}_{t-1}$ represents the cell state at time step t and $t-1$ respectively, similarly $\mathbf{h}_t$ and $\mathbf{h}_{t-1}$ represents the hidden state output at time step t and $t-1$ respectively. $\mathbf{x}_t$ is the input at time t [Figure 5.1]. The input $\mathbf{x}_t$ and the previous hidden state output $\mathbf{h}_{t-1}$ are concatenated as $[\mathbf{h}_{t-1}, \mathbf{x}_t]$ and are multiplied by respective weights for each gate.

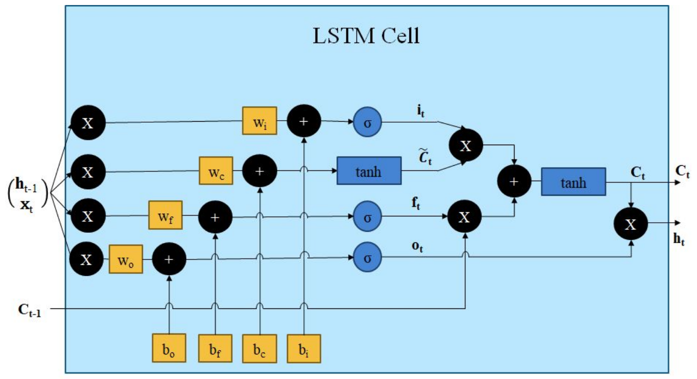


Figure 5.1: A pictorial representation of LSTM Cell.

Forget gate takes two inputs i.e. $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$, representing the output at time $t-1$, and input at time t, respectively. The vectors $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ are concatenated to form a vector $[\mathbf{h}_{t-1}, \mathbf{x}_t]$ and is multiplied by the forget weight matrix $\mathbf{W}_f$. The equation governing this operation is given by equation 5.1.

$$\mathbf{f}_t = \sigma(\mathbf{W}_f.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f). \tag{5.1}$$

Here, $\mathbf{f}_t$ represents the forget gate output and $\mathbf{b}_f$ is the forget gate bias. The forget gate, decides the amount of information to retain and it will forget the remaining information.

Next, the input gate decides the amount of information addition to the cell state. The concatenated vector $[\mathbf{h}_{t-1}, \mathbf{x}_t]$ is passed to the sigmoid function, and the tanh activation function simultaneously. The sigmoid gate represents the input gate [equation 5.2] and the tanh gate represents the cell state gate [equation 5.3].

$$\mathbf{i}_t = \sigma(\mathbf{W}_i.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i), \tag{5.2}$$

where, $\mathbf{W}_i$ represents the input gate weights, $\mathbf{i}_t$ represents the input gate output and $\mathbf{b}_i$ is the input gate bias. The input gate, decides the amount of information that would be updated.

$$\bar{\mathbf{C}}_t = tanh(\mathbf{W}_C.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C). \tag{5.3}$$

Here, $\mathbf{W}_C$ represents the cell state gate weights, $\bar{\mathbf{C}}_t$ represents the candidate cell state gate vector and $\mathbf{b}_C$ is the cell state gate bias. In the cell state gate, a vector of new candidate values $\bar{\mathbf{C}}_t$ are created, which would be added to the final cell state.

The final cell state is given by equation 5.4.

$$\mathbf{C}_t = \mathbf{f}_t.\mathbf{C}_{t-1} + \mathbf{i}_t.\bar{\mathbf{C}}_t. \tag{5.4}$$

The cell state transfers relevant information down the sequence processing. Thus, it can learn long-term dependencies. Now, the concern is about which filtered part of the cell state is needed to go to the output. It is decided by the output gate result $\mathbf{o}_t$, given by,

$$\mathbf{o}_t = \sigma(\mathbf{W}_o.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o). \tag{5.5}$$

Here, $\mathbf{W}_o$ represents the output gate weights and $\mathbf{b}_o$ is the output gate bias.

The final hidden state output thus can be written as,

$$\mathbf{h}_t = \mathbf{o}_t.tanh(\mathbf{C}_t). \tag{5.6}$$

Using these four gates, the LSTM can decide which information to forget and which information to retain down the memory and thus can be used to model time-series data efficiently.

5.3.2 Open-High-Low-Close (OHLC) Constraints

Daily stock prices are often represented in terms of four variables namely opening, high, low and closing prices. The names of these variables indicate straightforward price values. Since the stock prices vary in real-time, the data of a day is stored with respect to how high the stock price has been (high price), how low it has been (low price), at what price of the stock the market opened (open price) and at what price of the stock the market closed (close price). The open price of a particular stock is determined by the demand and supply at the start of the market and close price is the final price at which the stock is sold for the day. The Bombay Stock Exchange usually opens at 9.15 a.m. and closes at 3.30 p.m. Thus, the maximum value that the stock has attained within this time is called the high price and the minimum value that the stock has attained within this time is called the low price.

Thus, there are some clear indications as to what kind of constraints these variables should have. Then, it is obvious that the high price will have the highest value and the low price will the lowest value among these four variables. The open and close price should lie somewhere in between the former two extremal values. Let us denote the open, high, low and close prices of a particular stock as X_O, X_H, X_L and X_C. The following constraints exist which should be maintained during prediction tasks.

(i) The values of the prices must not be negative or zero.

$$X_O > 0, \tag{5.7}$$

$$X_H > 0, \tag{5.8}$$

$$X_L > 0, and \tag{5.9}$$

$$X_C > 0. \tag{5.10}$$

(ii) The high price should be greater than the low price.

$$X_H > X_L \tag{5.11}$$

(iii) The open and close price should lie in between the high and the low price.

$$X_O \in [X_H, X_L], and \tag{5.12}$$

$$X_C \in [X_H, X_L]. \tag{5.13}$$

These are also often represented as a Japanese candlestick chart [43] [Figure 5.2].

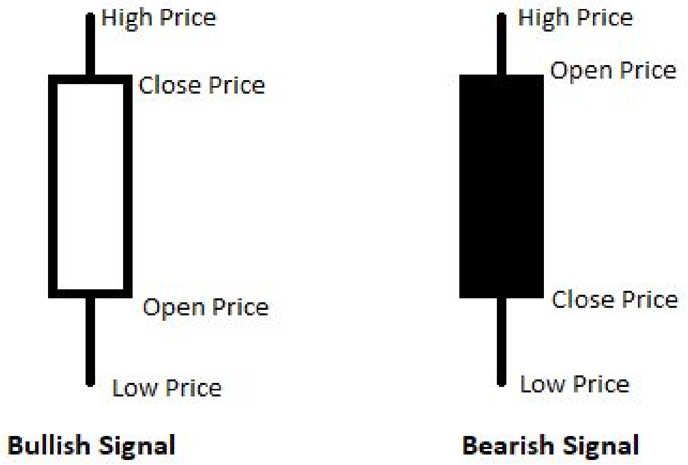


Figure 5.2: Pictorial representation of Japanese candlestick chart for OHLC data.

When the body is solid or filed in black, it indicates a bearish signal and when it is hollow or white, it indicates a bullish signal. There are many interesting patterns which can be found in such OHLC based charts and often depict emotions of the people surrounding a stock and are hence much informative than other trading indicators.

5.3.3 Multi-task Learning

While we are training a ML model for a certain task, the traditional method is to train a single model or an ensemble till its performance no longer improves. This makes the model to focussed on that specific task and so, we often ignore how this model can be utilised to learn for some related tasks. Transfer Learning [190] has been explored in the attempt to learn related tasks. However, such tasks are often independent of each other and mostly belong to different domains. So, they can utilise the same optimisation function for both the tasks. Multi-task learning [185, 191] (MTL) enables sharing of representations which takes into account the relationships between the related tasks, even if they are constrained. MTL models have two main types of layers when it comes to neural networks: shared layers and task specific layers [Figure 5.3]. The shared layers

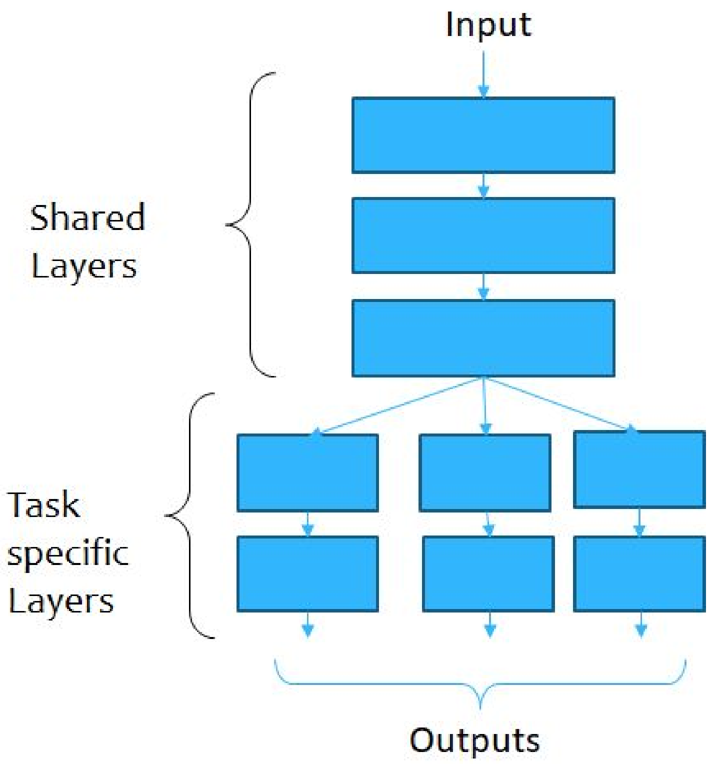


Figure 5.3: Pictorial representation of an multi-task learning model with shared and task-specific layers.

perform a sort of inductive transfer of information across the multiple tasks and the task-specific layers allow the independent learning of tasks. In the present case, we have the tasks of predicting the OHLC values which may seem to behave independently on a macro level (as the day's opening price cannot judge what the lowest or highest will be), but are constrained in a micro level (as described in section 5.3.2). MTL allows a restrained independence by drawing insights from it. This particular property allows an MTL model to be prone to overfitting as it is trying to capture the representations of all the tasks simultaneously. It has been shown [186] that as the number of tasks grows, the risk of overfitting the shared parameters becomes less than the risk of overfitting the task-specific layers.

5.4 Proposed Method

The proposed method takes advantage of the feature extracting capabilities of the autoencoder and the predictive abilities of LSTM network. The experiments show that using a single-task learning model to predict OHLC prices is not enough to have profitable recommendations for investable stocks. So, a MTL model is proposed in this regard. The entire method deals with prediction of OHLC prices of stocks and then recommending the relevant stocks to the user. It is observed that the proposed method does not result in a single loss over a test period of 350 days if the top profitable recommended stocks are actually invested in. The following segments describe the datasets and the proposed method in detail.

5.4.1 Dataset Description

In this analysis twenty seven Indian companies' stocks are considered which are listed by Fortune India 500. The data has been taken from the Bombay Stock Exchange website (`www.bseindia.com`). The datasets contain the open, high, low and close prices of the 27 stocks from the date 01-01-2014 to date 01-01-2019. There are several days in between this date range when the stock market was closed due to holidays. So, the effective number of samples were only available for 1234 days. The last 350 days (about 30%) from 02-08-2017 to 01-01-2019 for all the companies are taken as the testing period and the former days (from 01-01-2014 to 01-08-2017) are taken for training (around 70%) and validation of the model.

5.4.2 Proposed Normalisation Strategy

The main restriction to stock-price modelling lies in the normalisation strategy. For any neural network, a proper normalisation always offers benefits [192]. It has been show [193] that since stock prices have a wide range of values, normalisation of the prices to a fixed scale improves the prediction capabilities of deep RNNs. In this context, several normalisation methods have been used in the literature. Since the prediction models often take a window of past values into account to predict the next day's value, most of the methods till rely on the min-max normalisation of the prices over that window. For example, if a stock is studied for 100 days, its price may fluctuate from the order of tens to the order of thousands within a few days. If the window is chosen to be 20, then one would get 81 overlapping windows of data each of which will have a desired prediction value for training. And each of these 81 windows will be individually normalised using min-max normalisation. So, the price value which is the highest among each window will be mapped to 1 and the lowest will be mapped to zero. This implies that the same value may get different mappings if the window contains different values. For training data, this normalisation is easy. But, for data where the desired prediction value is not present, or for days when the prediction value makes a sudden jump from a very low range to a very high range or from a high range to a very low range, the predicted value should ideally be more than 1 or less than 0 respectively. In that case, de-normalisation becomes faulty. So, a proper normalisation strategy is important which will handle all the situations and all possible values of prices. A sigmoid-like function is much desirable in such cases where the values $[-\infty, \infty]$ is mapped to $[0, 1]$. However, using sigmoid straightaway is again problematic as any value greater than two will be mapped near to 1. So, the values in the order of hundreds or thousands will not be as much differentiable from one another. Now, one may utilise the properties of OHLC prices (that they cannot be negative or zero), and construct a logarithmic alternative. Thus, the proposed normalisation strategy is given as,

$$y' = \frac{1}{(1 + e^{-z})}, \tag{5.14}$$

where,

$$z = log_{10}(1 + x). \tag{5.15}$$

Then, the de-normalisation is straightforward given by,

$$x = 10^{(-ln(\frac{1}{y'}-1)} - 1, \tag{5.16}$$

The factor 1 is added to x to avoid any computational restrictions for very low (>0) values of x. This normalisation is independent of any window operation and may be applied to any unknown price ranges as well. It can be proven that if the proposed normalisation is applied to OHLC data, the normalised values are also OHLC in nature. The normalised values lie in the range $(0.5, 1]$ for x in range $(0, 1]$. Thus, for OHLC, the values X_O, X_H, X_L and X_C being greater than zero, the normalise values are also greater than zero. Moreover, since the sigmoid function and the logarithmic function is monotonically increasing for x>0, then the other constraints of OHLC also hold which makes the normalised value of X_H the highest and the normalised value of X_L the lowest. The normalised values of X_C and X_O will lie between the normalised value of X_H and X_L respectively. Let us denote the normalised value of X_O, X_H, X_L and X_C as y'_O, y'_H, y'_L and y'_C. These normalised values capture the variation of price ranges with respect to time. However, we need one extra set of features from the OHLC data, which would capture the variation of the prices with respect to each other without any temporal component to them.

We can construct another normalisation such that for prices X_O, X_H, X_L and X_C, the normalised prices are given by,

$$y_O = \frac{X_O - X_L}{X_H - X_L}, \tag{5.17}$$

$$y_H = \frac{X_H - X_L}{X_H}, \tag{5.18}$$

$$y_L = \epsilon, and \tag{5.19}$$

$$y_C = \frac{X_C - X_L}{X_H - X_L}. \tag{5.20}$$

The values y_O, y_H, y_L and y_C all lie in the range $(0, 1]$. This is equivalent to row-wise normalisation of the variables, which incorporate inter-variable relationships.

Thus, the proposed method uses eight variables $\{y'_O, y'_H, y'_L, y'_C, y_O, y_H, y_L, y_C\}$ for each date as input to the model.

5.4.3 Model Architecture

The proposed model consists of an MTL model consisting of LSTM layers preceded by an autoencoder's encoder pre-trained on the prices of the training days to reconstruct the input. The encoder of the pre-trained autoencoder is cascaded before the input of the proposed MTL model. There was a single hidden layer in the autoencoder with 4 hidden units. The number of hidden units for the AE network was selected using Shibata Ikeda's method [194]. The architecture of the AE used is given in Figure 5.4. After the AE is trained, the decoder is discarded and the encoder is used to extract the features from the eight variables of each date. These features are then fed to the predictor network for further prediction. The predictor network is constructed in a MTL fashion having some shared and some task-specific layers. It should be noted that the encoder is also a part of the shared layers if the entire model is considered. A pictorial representation of the entire model is given in Figure 5.5. For predictive tasks, the input is often a series of feature values from which the future values of certain variables are sought as output.

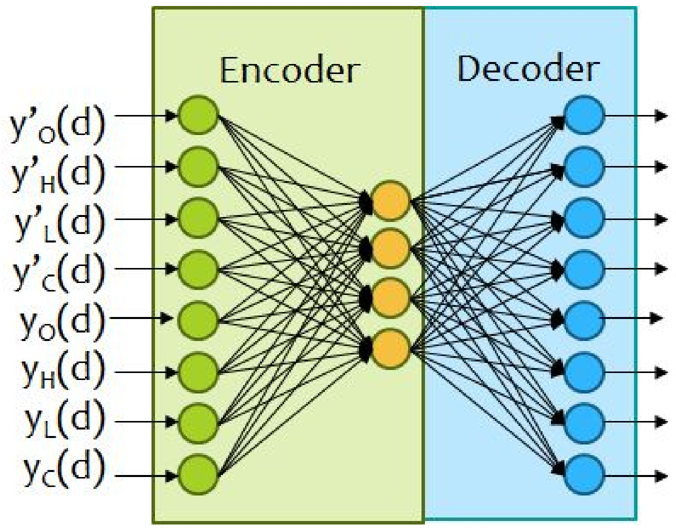


Figure 5.4: Pictorial representation of the AE network used in the proposed method.

Hence, for the present multi-task learning setting, the initial layers should be shared by the individual task (each task in the given scenario corresponds to predicting one of the output values). The shared layers would learn the inter-relationships between the input variables and pass it to the task specific layers which would, in turn, learn to predict the output values independently as a separate task.

The choice of the number of shared layers and task specific layers have been done using cross-validation and is elaborated in section 5.5. Since the predictions need to be made on the next day's OHLC values, one may predict any one of y'_O, y'_H, y'_L and y'_C to get the range of values and rest three of y_O, y_H, y_L and y_C for finding the inter-variable relationship for the next day. The last dense layers consists of sigmoid activation function. Thus, the output prediction is always in the range $[0, 1]$ as desired. In the proposed model, the four variables chosen to be predicted are y'_L, y_O, y_H and y_C. Let the predicted values of the four variables are y'^{Pred}_L, y^{Pred}_O, y^{Pred}_H and y^{Pred}_C. It was seen by experiments (section 5.5) that the low price is less volatile than all the other prices. So, predicting the range of values based on the low price would be much more suitable and other price values may be rebuilt using that. Thus, the actual predicted values of prices

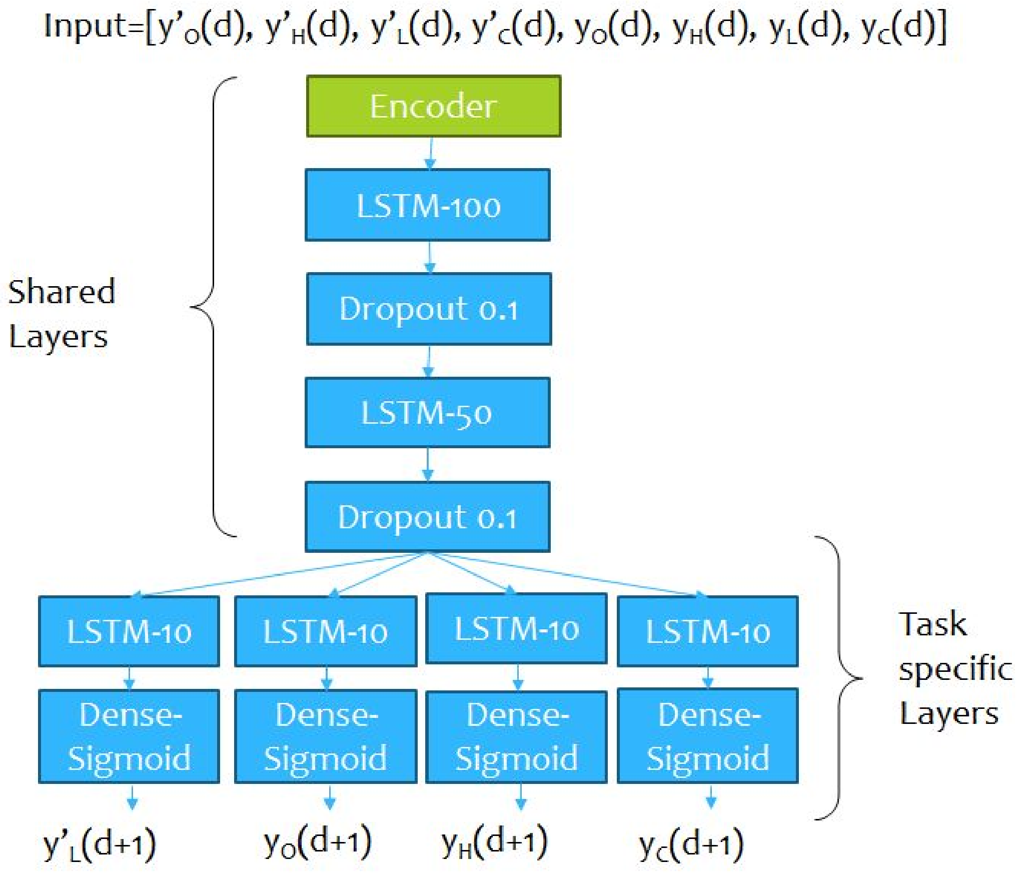


Figure 5.5: Pictorial representation of the proposed multi-task learning model.

may be rebuilt using the following equations,

$$X_L^{Pred} = 10^{\left(-ln(\frac{1}{y_L'^{Pred}}-1)\right)} - 1, \tag{5.21}$$

$$X_H^{Pred} = \frac{(1 - y_H^{Pred})}{X_L^{Pred}} \tag{5.22}$$

$$X_O^{Pred} = y_O^{Pred}.(X_H^{Pred} - X_L^{Pred}) + X_L^{Pred} \tag{5.23}$$

$$X_C^{Pred} = y_C^{Pred}.(X_H^{Pred} - X_L^{Pred}) + X_L^{Pred} \tag{5.24}$$

Thus, by default these predicted values maintain the OHLC constraints.

The entire model is trained using a sliding window [195] based backpropagation. A look-back period or window size of 20 days is taken to predict the next day's prices. For the training data, the predicted day's values are matched with the actual values for that day and the model is trained to reduce the mean squared error of each price independently in the task specific layers. For the testing days, the past 20 days' values are used to predict the next day's values.

5.4.4 Recommendations

After the training, the Opening Price, Closing Price, High Price and Low Price for the next day are predicted for each stock, taking the last 20 days' features as input. Since the proposed method considers only intra-day trading scheme, we are only concerned about the behaviour of the stocks on the next day. As the behaviour is predicted one day ahead, the stocks that the user may buy the next day can be efficiently recommended. Using the predicted data for the i^th day, two indicator variables namely Profit and William's %R [196] are constructed for the i^th predicted day.

Let us use the abbreviations where, $X_O^{Pred}(i)$ indicate the predicted opening price, $X_H^{Pred}(i)$ indicate the predicted highest price, $X_L^{Pred}(i)$ indicate the predicted lowest price and $X_C^{Pred}(i)$ indicate the predicted closing price on the i^{th} day. Similarly, $X_O^{Actual}(i)$

indicate the actual opening price, $X_H^{Actual}(i)$ indicate the actual highest price, $X_L^{Actual}(i)$ indicate the actual lowest price and $X_C^{Actual}(i)$ indicate the actual closing price on the i^{th} day.

It is quite intuitive that the maximum profit one can attain in a particular stock is when he/ she buys the stock on the previous day $i-1$ (when the price was the lowest i.e. at $X_L^{Actual}(i-1)$) and holds it for the next day i till it achieves a highest price ($X_H^{Actual}(i)$) will be given as equation 5.25.

$$ActualProfit = (X_H^{Actual}(i) - X_L^{Actual}(i-1)), \tag{5.25}$$

So, one may sell closer to the predicted highest price on i^{th} day to attain a profit close to the actual profit. The profit obtained by the person by following the proposed model's recommendation is given by equation 5.26.

$$Profit = (X_H^{Pred}(i) - X_L^{Actual}(i-1)), \tag{5.26}$$

Another important momentum indicator used by traders is the William's %R. It is used to know about the stocks which a user should buy. Its value oscillates between values 0 to -100. It is given as equation 5.27.

$$William's\%R = \frac{(X_H(Max) - X_C^{Pred}(i)}{(X_H(Max) - X_L(Min))} \times 100\%, \tag{5.27}$$

Here, $X_H(Max)$ signifies the highest high price obtained in the recent d days. The value of d is usually chosen to be 14 by the stock analysts. In order to choose the $X_H(Max)$, the past $d-1$ days of actual high price data is present. So, including the predicted high price $X_H^{Pred}(i)$ in consideration would make more sense to calculate the predicted value of William's %R. So, $X_H(Max)$ is given as,

$$X_H(Max) = max(X_H^{Actual}(i-d), X_H^{Actual}(i-d+1), X_H^{Actual}(i-d+2), \\ ...X_H^{Actual}(i-1), X_H^{Pred}(i)) \quad (5.28)$$

Also, $X_L(Min)$ signifies the lowest low price achieved in the recent d days. Since the actual data for $d-1$ days is present and one can only predict the lowest price of the next day ($X_L^{Pred}(i)$), $X_L(Min)$ is calculated as,

$$X_L(Min) = min(X_L^{Actual}(i-d), X_L^{Actual}(i-d+1), X_L^{Actual}(i-d+2), \\ ...X_L^{Actual}(i-1), X_L^{Pred}(i)). \quad (5.29)$$

It is indicated that a William %R mark below -80 is considered to be an oversold case [154] and suggests a buying signal.

In order to test the proposed model's efficacy, we are recommending stocks in two phases:

(i) Highest profitable stocks: From the list of the profits generated by each stock on a particular day, the top company is chosen, which can maximize the predicted amount of profit on the next day.

(ii) Stocks which will be overbought: The predicted William's %R is calculated and if any stock crosses the -80 mark (which suggests an overbought case), it is recommended.

The following section describes the various experiments conducted and results obtained for all the 27 Indian companies' stocks.

5.5 Experimental results

There are several aspects of experimentation as the proposed method tries to handle the problem of stock price prediction from various perspectives. The first experiment reveals why the proposed normalised low price ($y_L'^{Pred}$) was chosen to be used as a prediction variable. The second experiment looks into various existing models and checks whether those really make meaningful predictions or not. The third experiment shows a comparison of various methods

5.5.1 Low Price is Less Volatile

It was shown by Gorenc and Velušček [164], that high price is less volatile than closing price. However, their experiments did not take the low and opening price into their consideration. Following the same argument, if we plot the normalised standard deviations of the various prices of the companies for the entire training period data, it is seen that the low prices is the least volatile [Figure 5.6] for most companies.

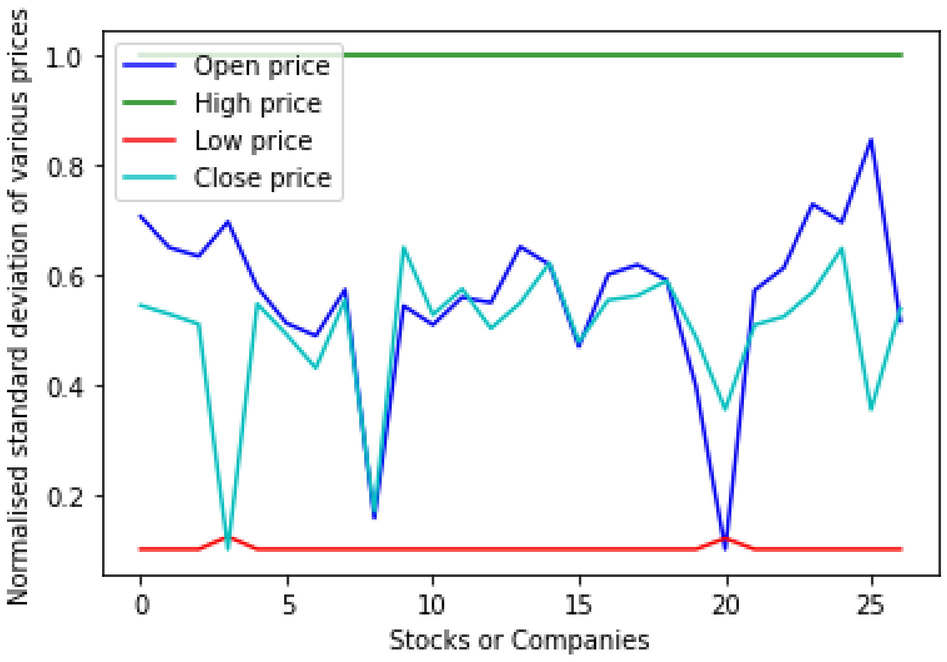


Figure 5.6: Graph showing the volatility of different prices.

The standard deviation is normalised to a range 0 to 1. Normalised standard deviation has been used in this case because the different companies had different price ranges and required mapping to a same range to view the effective results. It can be seen that the high prices have the highest volatility (highest standard deviation), which contradicts the earlier findings. This is mainly due to the fact that the authors used a window based normalisation of OHLC features prior to calculating the standard deviation which contributed to the faulty interpretation.

5.5.2 How do Autoencoders Benefit in Stock Data?

AEs have a tendency to map similar data to similar ranges. Now, if a particular stock has drastic value fluctuations within a very short period of time, the actual OHLC feature space would be overwhelmed with the disparity in values and will fail to capture the interesting trends in the data. This is analogous to a sudden event in the time. However, it cannot be directly termed as an anomaly as the state of increased or decreased value persist for a long time after it has started. This has been observed for many stocks in the dataset. One such stock is of Reliance India Limited, where in the test days, the value of the stock fell drastically (to almost half the value in a single day) as seen in Figure 5.7 (marked in red).

The original OHLC feature space only captured the difference between values, but the interesting relationships between the prices were completely lost. As seen in figure 5.8, none of the four OHLC features could show any difference between the bull or bear days. However, AE was not so overwhelmed with the difference in prices and the four features extracted by AE could easily differentiate between the bull and bear days (Figure 5.9).

5.5.3 Existing Methods Fail to Capture OHLC Constraints

The proposed method used rolling windows model for all the RNN based models. The predicted prices are a stock's open price, close price, high price, low price for the next

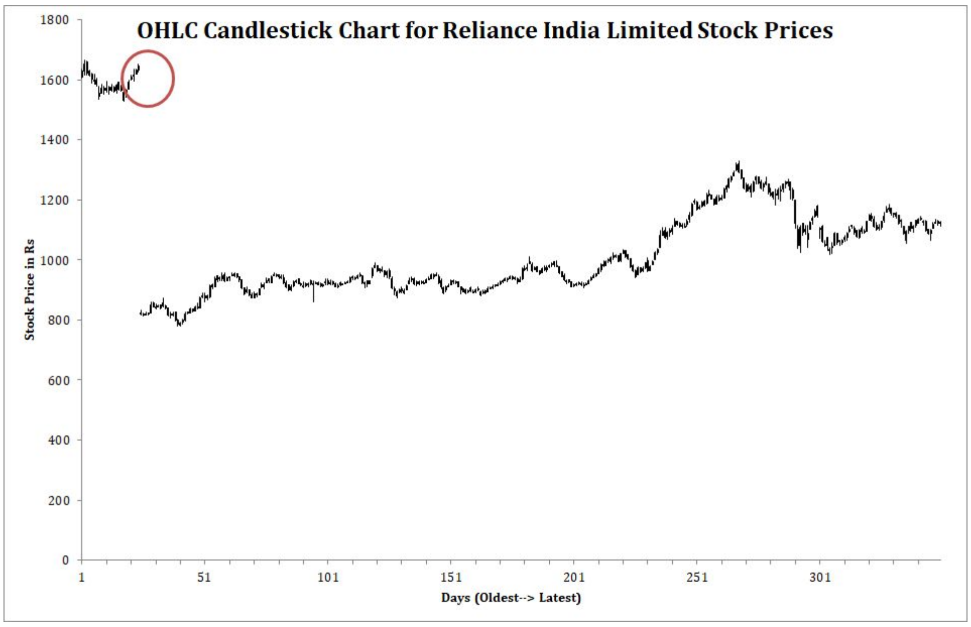
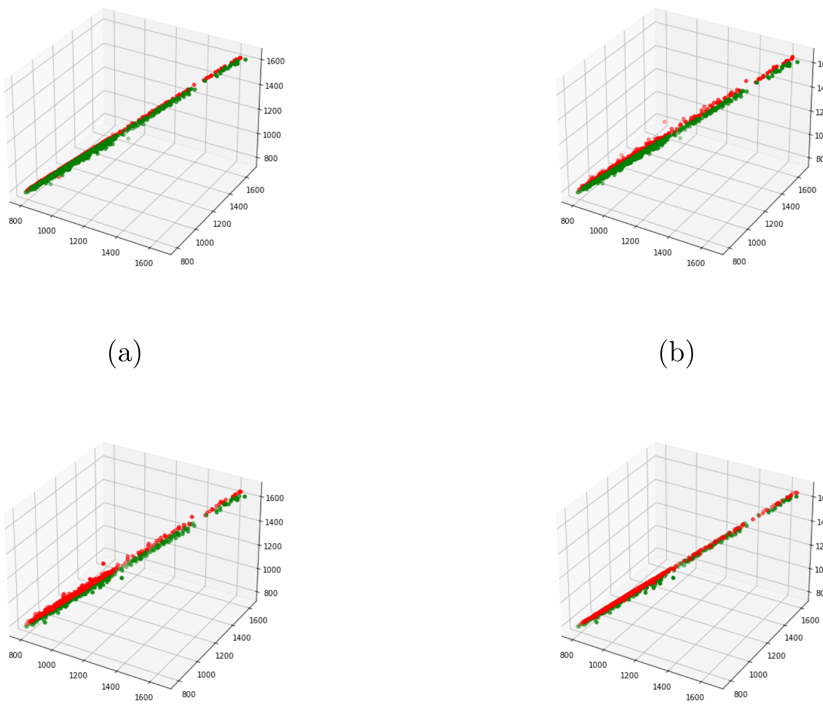
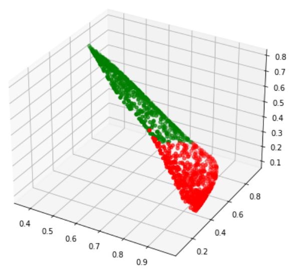


Figure 5.7: OHLC candlestick chart for Reliance India Limited for the test days. (The point of drastic decrease in price has been marked by red in the figure.)

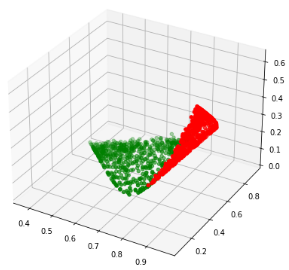


(a) (b) (c) (d)

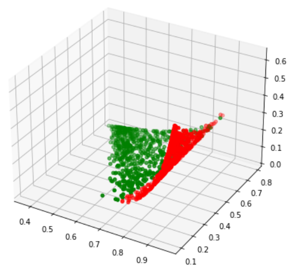
Figure 5.8: Plot of four original OHLC features with respect to one another. (Green points represent Bullish days and red points represent Bearish days.) (a) Features O, H and L, (b) Features O, H and C, (c) Features O, L and C, (d) Features H, L and C.



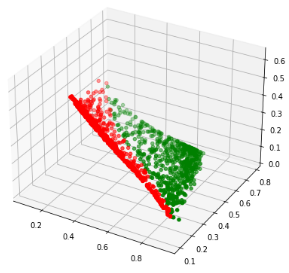
(a)



(b)



(c)



(d)

Figure 5.9: Plot of four AE extracted features with respect to one another. (Green points represent Bullish days and red points represent Bearish days.) (a) Features 1, 2 and 3, (b) Features 1, 2 and 4, (c) Features 1, 3 and 4, (d) Features 2, 3 and 4.

day. In all the cases a separate network is trained for each company in the dataset. Since the existing models used for stock price prediction do not take into account the OHLC constraints, we have compared the proposed method with only those models which predicted proper OHLC prices to keep the comparison fair. Although some existing methods [165, 185, 197]claim to predict OHLC, they fail to capture the constraints. In order to support our claim that existing methods fail to follow OHLC constraints, we have tested checked the predictions for all the 27 companies. If any of the constraints do not hold for any of the test days, we have increased the number of failure counts by one. The observations are given in Table 5.1.

Table 5.1: Number of test days when the existing methods fail to follow the OHLC constraints.

Name of Stock	Proposed	ARIMA[197]	MTL with BI-LSTM [185]	LSTM [165]
APSEZ	0	1	0	1
ASIAN PAINTS	0	0	1	0
AXIS BANK	0	2	1	2
BAJAJ AUTO	0	2	15	10
BAJAJ FINSERV	1	3	20	23
BPCL	0	1	1	2
AIRTEL	0	0	0	0
CIL	0	1	2	1
GAIL INDIA	0	2	0	0
HCL	0	2	2	3
HDFC BANK	0	1	2	7
HINDALCO	0	2	0	2
HP	0	0	1	1
ICICI BANK	0	4	18	12
ITC	0	0	0	4
IOCL	0	1	9	12
INFOSYS	0	1	0	1
KOTAK MAHINDRA BANK	0	0	0	0
L&T	0	1	3	2
M & M	0	0	0	1
NTPC	0	0	0	0
ONGC	0	2	0	0
PFC	0	0	0	0
POWERGRID	0	0	0	0
PNB	0	2	14	9
REL	0	0	5	17
RELIANCE	0	3	2	13

It can be seen that the proposed method maintains the OHLC constraints for all the companies' stocks for all the test days while the existing methods fail for multiple companies.

5.5.4 Comparison of Methods

It would be unfair to compare the proposed model with the methods which fail to follow OHLC constraints. Thus, standard models like ARIMA [197], VAR (vector autoregression) [198], SVR (support vector regression) [178], VEC (vector error correction) [198], etc which have no intrinsic ability to capture the OHLC constraints, were not considered. However, there are some methods which capture the OHLC constraints like the FLF-LSTM [184] (even though they explicitly do not handle it) and existing methods with LSTM [165] and Bi-LSTM based multi-task model [185] may be modified according to the proposed features and AE to predict the proposed outputs. Hence, the following methods have been compared:

(i) Proposed AE-MTL hybrid model with the proposed features as inputs. Let us denote it by PF-AE-PMTL.

(ii) Proposed MTL model without AE and just the proposed features as inputs. Let us denote it by PF-PMTL.

(iii) LSTM model [165] without AE and just the proposed features as inputs. Let us denote it by PF-LSTM.

(iv) LSTM model [165] with AE and the proposed features as inputs. Let us denote it by PF-AE-LSTM.

(v) Bi-LSTM based multi-task model [185] without AE and just the proposed features as inputs. Let us denote it by PF-BiLSTM-MTL. (This model cannot be hybridised with AE as the authors have assumed the initial layers to be task specific layers.)

(vi) FLF-LSTM [184] with the original non-normalised OHLC features (as the authors suggest that the method needs no normalisation.)

Figures 5.10 5.11, 5.12 and 5.13 show the root-mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE) and R-squared (R^2) value of different methods for various company's stocks averaged over 10 independent runs.

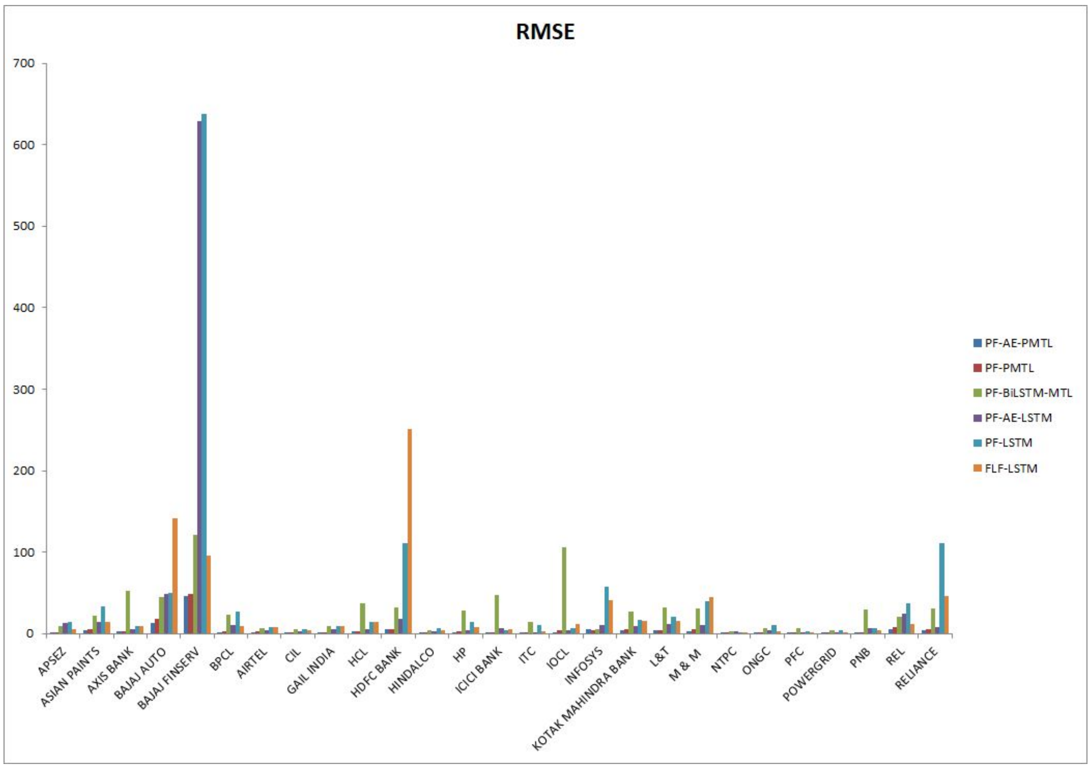


Figure 5.10: Comparison of root-mean squared error (RMSE) of different methods.

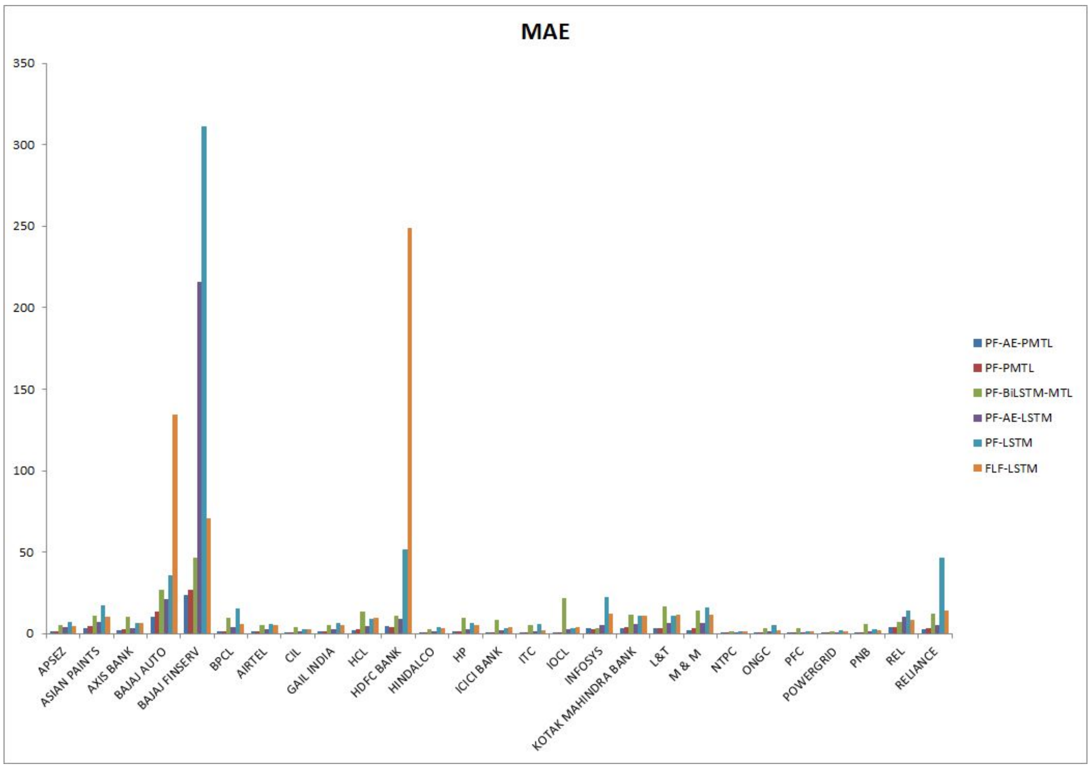


Figure 5.11: Comparison of mean absolute error (MAE) of different methods.

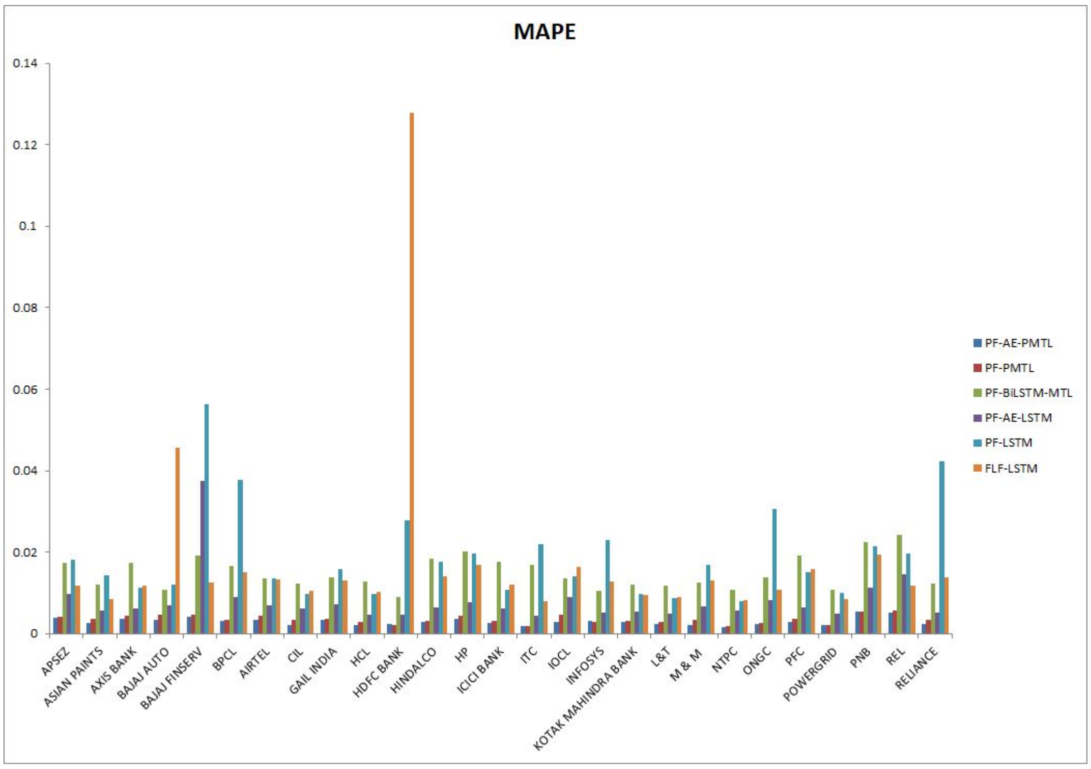


Figure 5.12: Comparison of mean absolute percentage error (MAPE) of different methods.

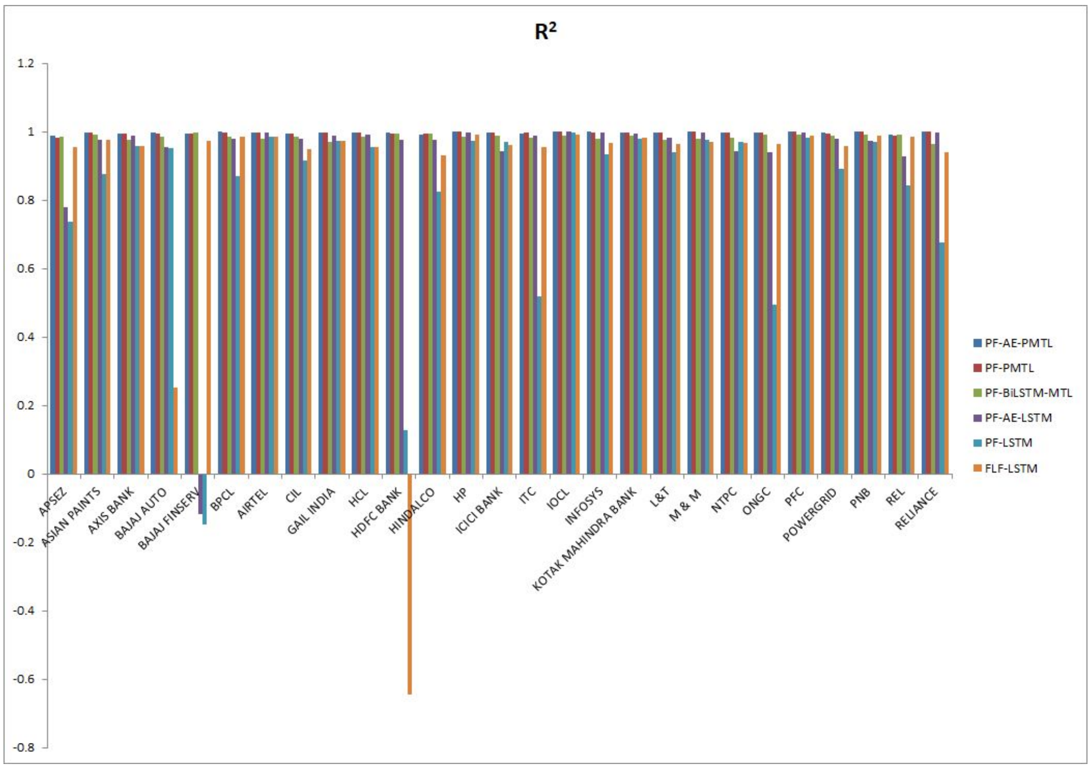


Figure 5.13: Comparison of R-squared (R^2) of different methods.

Let us assume that the original OHLC prices for the i^{th} day are denoted by $X_O^{Actual}(i)$, $X_H^{Actual}(i)$, $X_L^{Actual}(i)$ and $X_C^{Actual}(i)$, respectively and the predicted OHLC prices for the same i^{th} day are denoted by $X_O^{Pred}(i)$, $X_H^{Pred}(i)$, $X_L^{Pred}(i)$ and $X_C^{Pred}(i)$, respectively. Calculation for RMSE, MAE, MAPE and R^2 can be given by equations 5.30, 5.31, 5.32 amd 5.33.

$$RMSE = \frac{1}{N}\sum_{i=1}^{N}[(X_O^{Actual}(i) - X_O^{Pred}(i))^2 + (X_H^{Actual}(i) - X_H^{Pred}(i))^2 + (X_L^{Actual}(i) - X_L^{Pred}(i))^2 + (X_C^{Actual}(i) - X_C^{Pred}(i))^2]^{\frac{1}{2}}. \quad (5.30)$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}[|X_O^{Actual}(i) - X_O^{Pred}(i)| + |X_H^{Actual}(i) - X_H^{Pred}(i)| + |X_L^{Actual}(i) - X_L^{Pred}(i)| + |X_C^{Actual}(i) - X_C^{Pred}(i)|]. \quad (5.31)$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}[|\frac{X_O^{Actual}(i) - X_O^{Pred}(i)}{X_O^{Actual}(i)}| + |\frac{X_H^{Actual}(i) - X_H^{Pred}(i)}{X_H^{Actual}(i)}| + |\frac{X_L^{Actual}(i) - X_L^{Pred}(i)}{X_L^{Actual}(i)}| + |\frac{X_C^{Actual}(i) - X_C^{Pred}(i)}{X_C^{Actual}(i)}|]. \quad (5.32)$$

$$R^2 = 1 - \frac{SS_{Residual}}{SS_{Total}}, \quad (5.33)$$

where, the sum of squared residuals ($SS_{Residual}$) and total squared error (SS_{Total}) are given by,

$$SS_{Residual} = \sum_{i=1}^{N}[(X_O^{Actual}(i) - X_O^{Pred}(i)^2 + (X_H^{Actual}(i) - X_H^{Pred}(i))^2 + (X_L^{Actual}(i) - X_L^{Pred}(i))^2 + (X_C^{Actual}(i) - X_C^{Pred}(i))^2], \quad (5.34)$$

and,

$$SS_{Total} = \sum_{i=1}^{N}[(X_O^{Actual}(i) - mean(X_O^{Actual}))^2 + (X_H^{Actual}(i) - mean(X_H^{Actual}))^2 + (X_L^{Actual}(i) - mean(X_L^{Actual}))^2 + (X_C^{Actual}(i) - mean(X_C^{Actual}))^2]. \quad (5.35)$$

It can be seen that the proposed method (PF-AE-PMTL) has the lowest RMSE, MAE and MAPE for all the stocks. The proposed method has also consistently achieved a R^2 value close to 1 for all the stocks (a model that has R^2 value equal to 1 is a model with the best performance). Some of the methods have shown negative R^2 values indicating that the model does not capture the trend of the data at all. The results also show that the use of AE has clearly enhanced the performance of the proposed MTL model. It is evident from the fact that the RMSE, MAE and MAPE for PF-AE-PMTL is lesser than that of PF-PMTL for all the stocks. This shows that the proposed model is comparatively a better model than the other models in terms of predicting the stock prices accurately.

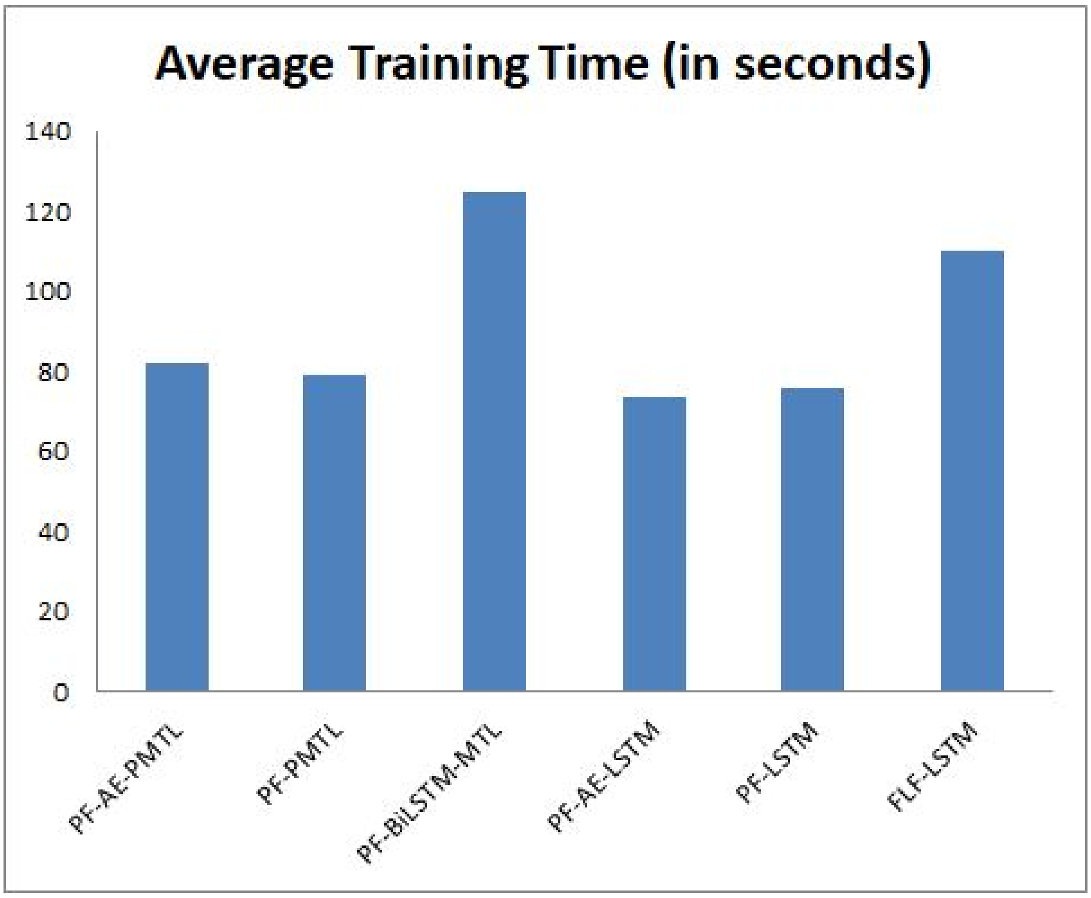
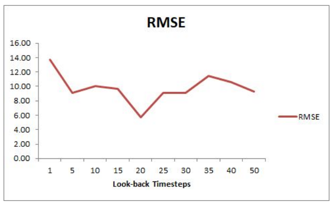


Figure 5.14: Comparison of the average training time of different methods.

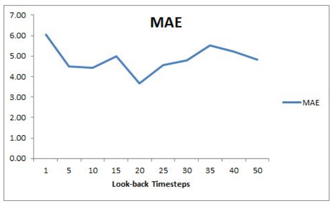
The average training time taken for training of each method is shown in Figure 5.14. The values are averaged over 10 independent runs of each method. It can be seen that the training time for the proposed method is comparable to the method with the lowest time for training (PF-AE-LSTM) and differs only by around 8 seconds. This shows that the proposed method is not so expensive in terms of training time.

5.5.5 Look-back Period or Window Size

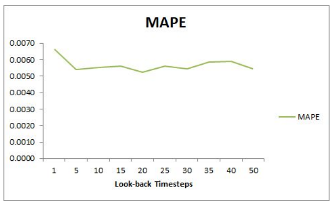
In order to know which look-back period was sufficient for training the proposed model, a validation data (10%) was separated out of the training days. This consisted of the last 123 days of the training days and the various parameters of the proposed model was validated on this particular chunk of the data. The RMSE, MAE and MAPE were



(a)



(b)



(c)

Figure 5.15: Plot of the (a)RMSE, (b)MAE and (c)MAPE for various look-back periods.

calculated for 10 independent runs of the model for various look-back periods upto 50 days for the Reliance India Limited's stock. It can be seen [Figure 5.15] that the look-back period of 20 days was the best in terms of all the three metrics. Thus, the look-back period for 20 was chosen for the proposed method. Increasing the look-back period might have resulted in lesser error rates, but increasing the look-back period would imply a higher dimensional input and hence increased complexity.

5.5.6 Number of Shared and Task-Specific Layers

Finding the number of shared and task-specific layers is a crucial task for the MTL models in general. Thus, various experiments were conducted to fix the architecture of the proposed MTL model. Since the model already had an encoder layer and a final dense layer is needed for the output, the number of shared and task-specific layers indicate the hidden layers in between. The experiments were conducted on the validation data as used in section 5.5.5.

Table 5.2: The RMSE, MAE and MAPE values for different number of shared layers in the model for the Reliance India Limited's stock.

Number of Shared Layers	**RMSE**	**MAE**	**MAPE**
0	24.65	15.29	0.0133
1	15.38	5.51	0.0070
2	5.72	3.66	0.0052
3	10.29	5.11	0.0057
4	7.03	4.46	0.0054
5	12.81	4.6	0.0058

Table 5.3: The RMSE, MAE and MAPE values for different number of task-specific layers in the model for the Reliance India Limited's stock.

Number of task-specific layers	**RMSE**	**MAE**	**MAPE**
0	258.77	162.78	0.1498
1	5.72	3.66	0.0052
2	17.46	4.49	0.0076
3	9.80	4.05	0.0066
4	21.08	4.99	0.0094
5	9.08	3.79	0.0062

It can be seen from tables 5.2 and 5.3 that having two shared layers and one task-specific layers resulted in the least error. It is to be noted that the This is analogous to the fact that too much depth in a DNN is often not good.

5.5.7 Recommender system

In order to show the applicability of the proposed model as a deployable solution, the proposed model has been used to recommend stocks which would prove to be the most profitable next day. In order to do so, $Profit$ was calculated as given in equation 5.26 and based on that the top profitable stocks were recommended. The recommender system would predict the high price for the next day and would recommend the stocks for which the net profit per share is the maximum. Since the $Profit$ (from equation 5.26) is a predicted value and holds no practical relevance, to show the efficacy of the model, the $ActualProfit$ (as calculated by equation 5.25), was plotted against each of the test days [Figures 5.16 - 5.21]. The intuition is that a good recommendation model will cause lesser (or almost no) losses on any of the test days. It can be seen from Figures 5.16, 5.17, 5.18 and 5.20 that all the other methods had multiple occasions in the test days where the $ActualProfit$ was negative (which indicates a loss). However, as seen from Figure 5.21, the proposed model had not made a loss for even a single day for all the test days.

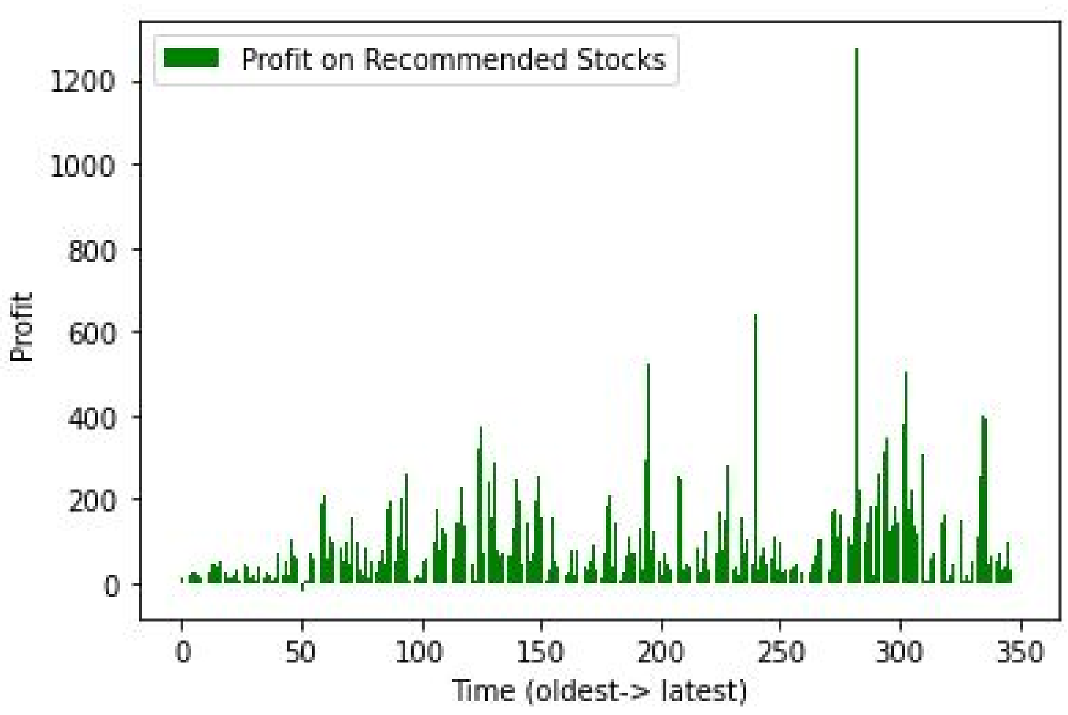


Figure 5.16: Actual profit obtained by the recommendations of the model PF-LSTM.

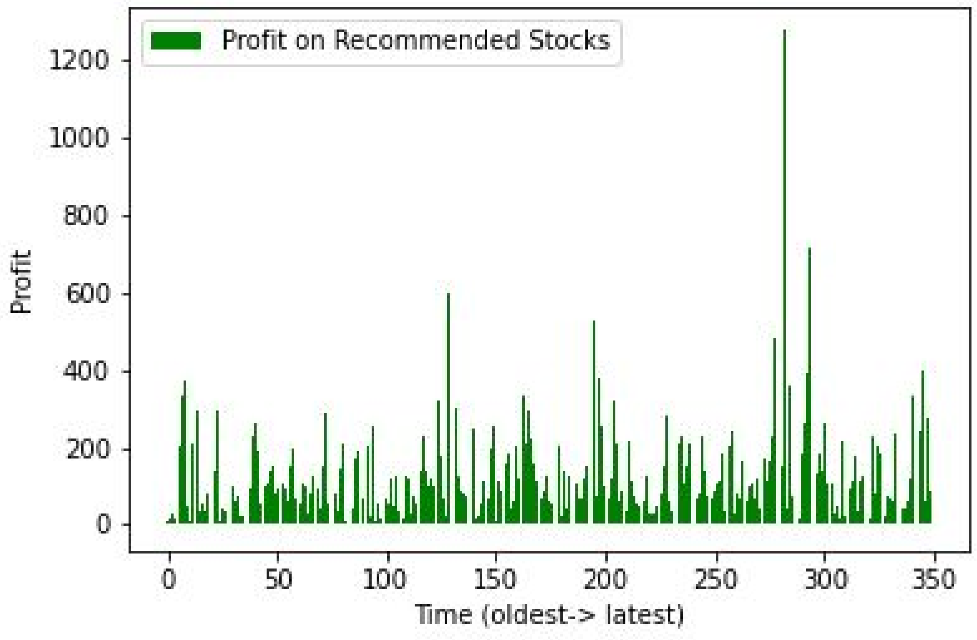


Figure 5.17: Actual profit obtained by the recommendations of the model PF-AE-LSTM.

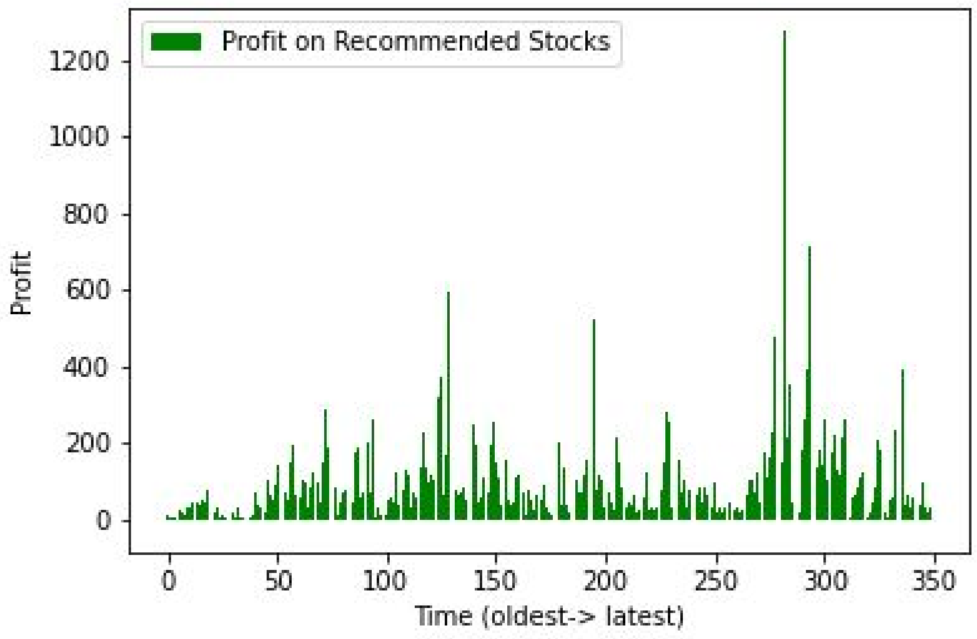


Figure 5.18: Actual profit obtained by the recommendations of the model PF-BiLSTM-MTL.

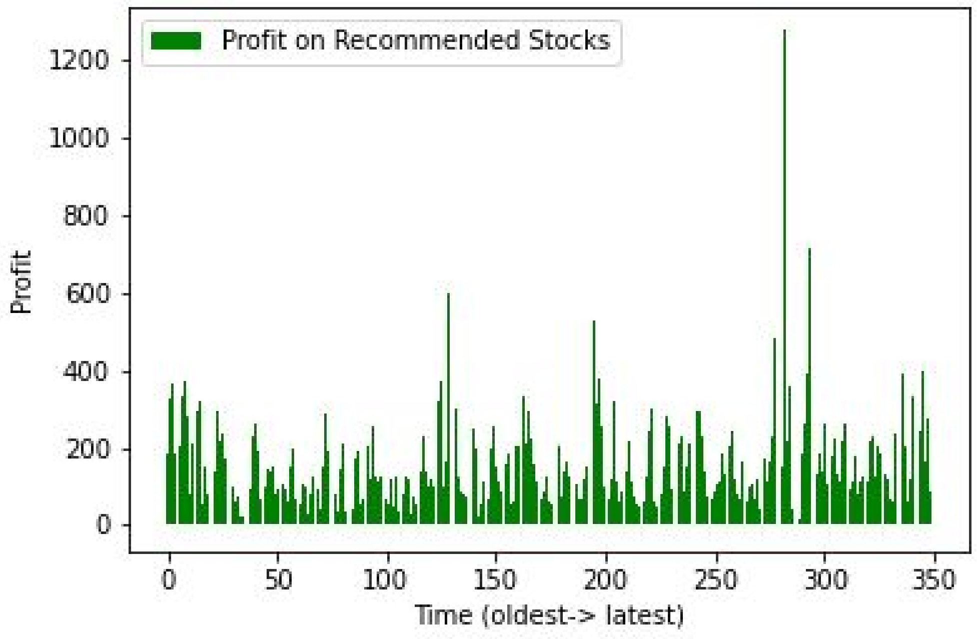


Figure 5.19: Actual profit obtained by the recommendations of the model FLF-LSTM.

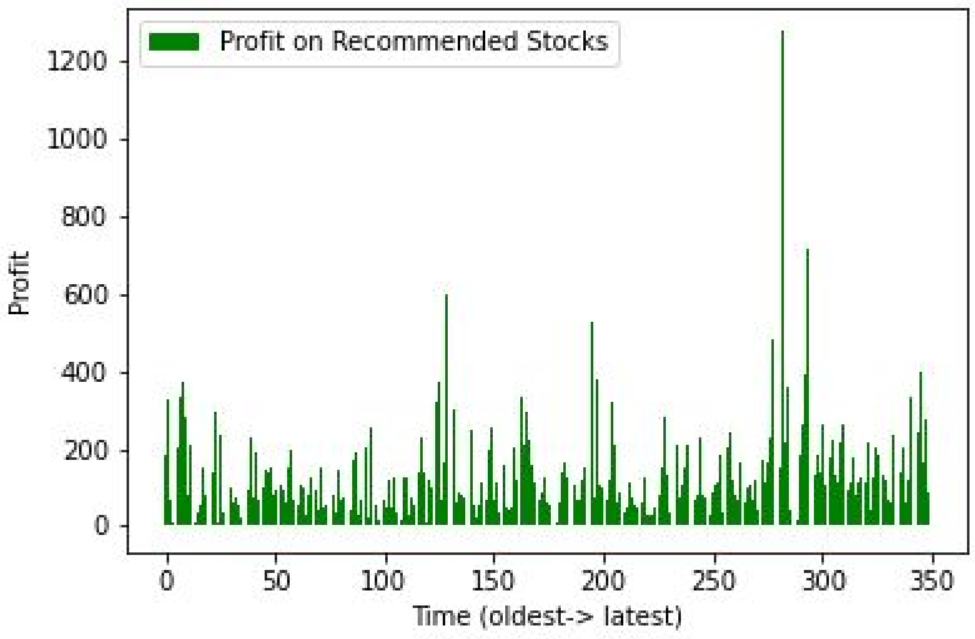


Figure 5.20: Actual profit obtained by the recommendations of the model PF-PMTL.

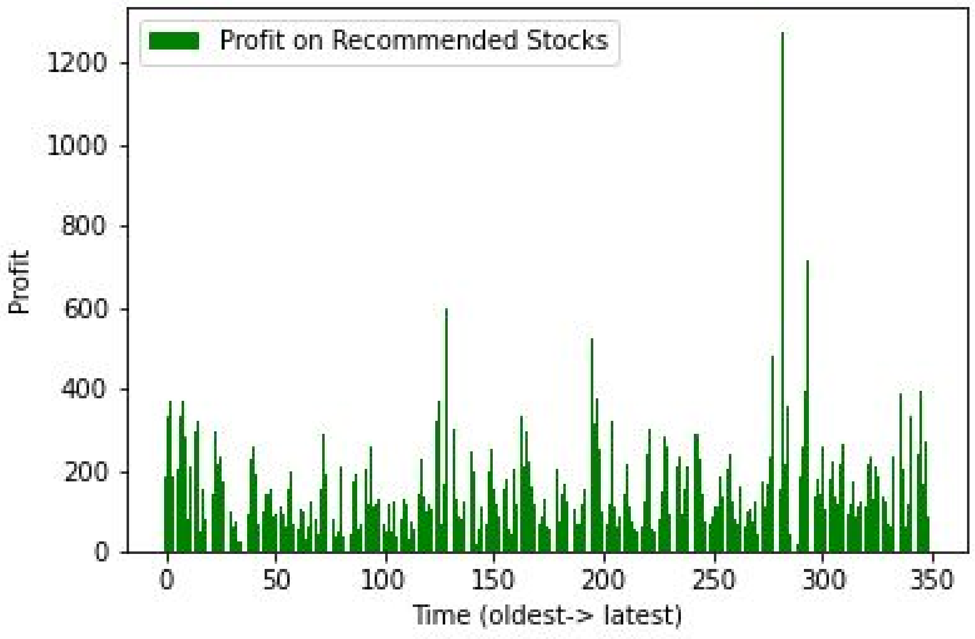


Figure 5.21: Actual profit obtained by the recommendations of the proposed model PF-AE-PMTL.

The next recommendation is based on William %R. This is a very important trend indicator for the investors. It is said that the investors usually buy the stock when the William %R is below -80. In order to give this particular type of recommendation, the proposed PF-AE-PMTL was analysed for ten random test dates.

Table 5.4: Comparison of actual and predicted William's %R for stocks which were recommended by the proposed model for buying.

Date	**Stocks**	**Predicted %R**	**Actual %R**
14-09-2017	RELIANCE	-98.39	-96.26
10-10-2017	L&T	-80.35	-88.09
15-11-2017	IOCL	-81.04	-92.94
22-02-2018	BAJAJ AUTO	-82.73	-97.15
	IOCL	-81.41	-96.61
	PNB	-87.41	-94.26
21-03-2018	IOCL	-93.60	-99.17
	PFC	-83.83	-78.88
23-03-2018	IOCL	-95.37	-98.70
25-04-2018	BPCL	-80.76	-81.34
	HP	-82.72	-98.49
	PNB	-80.16	-97.97
10-05-2018	CIL	-80.67	-88.55
	HCL	-87.00	-88.97
28-05-2018	RELIANCE	-83.89	-86.76
10-09-2018	BAJAJ FINSERV	-80.19	-94.50
	INFOSYS	-97.04	-98.01

It can be seen from Table 5.4 that most of the recommended stocks actually crossed the -80 mark of William %R on the five random test days. Only one recommended stock (PFC) did not cross the -80 mark on 21-03-2018, but the actual value of its William's %R was close to -80.

It can be seen from the experiments that the proposed model is quite efficient in making profitable recommendations to the users who follow the predictions for investing in stocks.

5.5.8 Autoencoders replaced by other feature extraction methods

It was evident that AE based projection of the proposed feature set was helpful in bringing out the inter-relationships between the variables (Section 5.5.2). But, it would be interesting to see if only AEs give this benefit or other projection methods too can give similar boost. Thus, an experiment was conducted by replacing the AE in the proposed model by different feature extraction methods like k-PCA, t-SNE, LLE, Isomap and GRP applied on the proposed feature set. Results are shown for the Reliance India Limited stocks because this particular stock had a sudden event of market crash as shown in Figure 5.7. It can be seen from Table 5.5 that the errors obtained by replacing AE with other

Table 5.5: The RMSE, MAE and MAPE values for the proposed PMTL model using different feature extraction methods for the Reliance India Limited's stock.

	RMSE	**MAE**	**MAPE**
AE	**6.528**	**3.838**	**0.004**
k-PCA	80.544	19.51	0.017
t-SNE	60.763	28.006	0.026
LLE	21.936	7.374	0.007
Isomap	54.14	9.537	0.008
GRP	84.065	11.37	0.009

feature extraction methods are substantially bigger in magnitude than what is obtained by using AE.

5.6 Conclusions and Future Work

This chapter presents an hybrid AE multi-task model for stock prediction. The work has been extended to build a recommender system. The inclusion of pre-trained encoder modules clearly helped in increasing the predictive capabilities of the proposed MTL model for OHLC predictions. The novelty of the proposed model lies in the unique normalisation strategy for OHLC data and explains why an AE might be better to use in problems like stock markets where the prices may fluctuate drastically. This angle of stock price prediction has mostly been overlooked in the literature. Thus, this work tries to address the gap of using ML in financial prediction tasks.

It is to be noted that the proposed work considers straightforward price values of stocks and does not consider any indicator variables used by the financial analysts. However, we are hoping to extend this work in future using other technical indicators as well. In this present analysis the information regarding the volume of stocks sold is not incorporated as the data for some companies had no information on volume. In future, we therefore wish to incorporate some volume indicators as well. We also hope to make a composite model in future which would combine both the fundamental as well as technical analysis for recommendation of stocks. The proposed models would have been more practical if the predictions could have given for very short-term or very long-term trading purposes, which is also one of the planned future directions.

Chapter 6

Autoencoder Pre-trained Distributed Deep Neural Networks for Outlier Detection

6.1 Introduction

In the former chapters, it is witnessed that AEs have been useful in extracting meaningful feature representations for datasets with imbalanced class distributions. Thus, it is intriguing to study how the problem extends to decentralised data settings. AEs have an innate ability to capture the outlier features (as seen in chapter 3). Now, the question arises whether the AE pre-initialisation [10] given to a DNN adds any further improvement to the capability of learning from imbalanced datasets. DNNs are inherently biased by the majority class and often fail to give good performance if the classes are severely imbalanced [36]. This problem is a big limitation when it comes to achieving high classification performances as compared to balanced classification tasks where DL thrives. DL in distributed setting [199] is a boon for the current times when we are facing the data deluge. Reports show that human beings have generated more data in just the last decade than has been produced over the last hundred years. This astonishing fact is seen as the foundation for several experiments of concurrent and distributed computation and distributed methodologies in computer science education. Distributed computation is a necessity of this age and it is the pinnacle of this area of science to combine it with sophisticated learning approaches such as DL (neural nets). However, distributed DL in an imbalanced scenario is a big gap that needs to be addressed. If the distributed model uses non-independent and non-identical (or simply non-i.i.d) chunks of data in each local computation server, the imbalance becomes critical and often too overwhelming for a DNN to handle. This is, however, a paradox as multiple experts are being used to model the classification boundary.

This may be viewed as an extension of the concept of using multiple classifiers to obtain a better classification performance. Such a setting is often regarded as Social Learning System (SLS) [200] or an Ensemble [201] or a Multi-Classifier System (MCS) [202]. This is due to the complementary nature of each of the classifiers in the SLS [200]. This diversity is usually achieved either through diversity in data or diversity in models. Deep neural networks under such ensemble settings have been proven to give better performance than their single model counterparts [201]. Perceiving that every independent classifier may make complementary errors, we can merge the decisions from all classifiers to build a composite framework that outperforms any single classifier. Such a combination of multiple classifiers can handle data complexity very well and supersede a single sophisticated classifier. Owing to the training on different inputs, the pool of predictors is ensured to be highly diverse and mutually complementary.

Borrowing the idea of these SLSs, it is expected that a distributed DL model should give better performance than their single model analogy. We can view this particular setting as an ensemble or a social learning system where multiple models in the nodes are trained on different parts of the data, and the aggregated weight update is performed in the main parameter server, it behaves like an SLS. This is also known as federated learning

(FL). In the parameter server version of the distributed setting the data, and samples are spread through a variety of processing nodes and each node has one deep network. There is a parameter server that receives all the parameters (weights and biases) after each communication round, aggregates them and sends the updated aggregated weights to all the worker nodes for the next communication round to begin [Figure 6.1]. This method of weight update often results in decreased performance as the number of predictors increases. This is mostly because the data distribution across the worker nodes does not have identical distributions and might also have local class imbalance issues. DNNs being just an extension of the traditional neural networks suffer from masking and swamping effects. However, DNNs are state-of-the-art models for most classification tasks. This problem is aggravated if the actual data has an inherent global imbalance (as in the case of outlier detection where the class imbalance is severe) and is so big that it needs distributed processing. As DNNs are inherently unsuitable for the problem of such outlier detection, distributed models fail. However, as seen in Chapter 3 and also identified by many researchers [16, 34], AEs are better suited as outlier detection models. Thus, it would be interesting to see how the AE pre-initialised DNNs perform for outlier detection tasks.

It is even more intriguing to study how an AE pre-initialised distributed architecture of DNNs using a global weight update mechanism at the parameter server would handle the severely imbalanced datasets as with the outlier detection problem. The global data is itself severely imbalanced (outlier class constitute $<10\%$ of the total samples) and thus the local datasets might be critically imbalanced (some may have a negligible percentage or completely lose samples from the minority class). Here arises Hace's theorem of Big Data [203] when the data is not globally available to all the worker nodes, each of the nodes train independently over any arbitrary portion of the data and makes predictions on that.

In a federated learning [204, 205, 206, 207] scenario, the local workers do not commu-

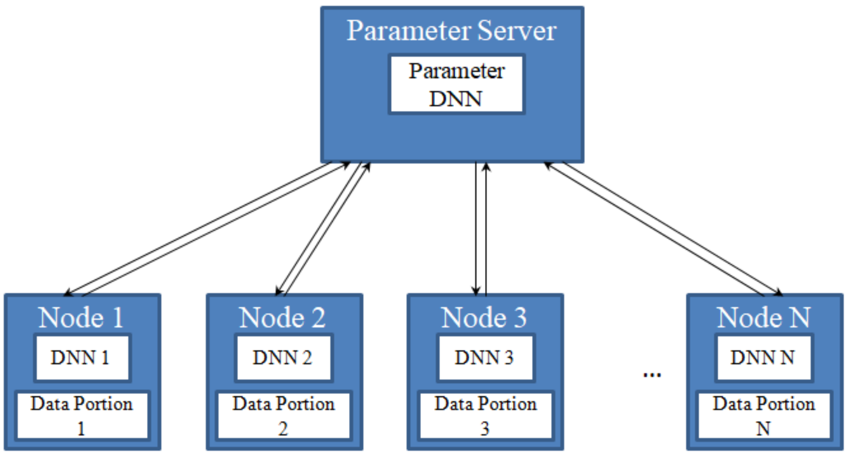


Figure 6.1: Schematic representation of a federated deep learning setting (DNN represents a deep neural network)

nicate among themselves and can only communicate indirectly via the parameter server where they only send the model parameters for aggregation. The parameter server does not contain any data for training. Training is only done in the worker nodes. Each worker node has one deep neural network (DNN) model and a portion of the entire dataset and all the neural networks of each worker node have identical architectures. The idea of FL is that each of the worker nodes does not reveal their local datasets to each other. This property is quite useful in scenarios where data privacy preservation is important among collaborating nodes. The parameter server has an identical deep neural network only for storing the final updated parameters after each communication round. The basic assumption for such FL implementations is that the data division across the worker nodes is independent and identically distributed (i.i.d) [208].

The traditional method, known as federated averaging (FedAvg) [209] works simply by averaging all the respective parameters from the worker nodes. This method works well in cases of i.i.d data sample division across worker nodes and is highly scalable. However, for the non-i.i.d setting, this method fails as the number of worker nodes increases [210]. Moreover, scenarios of class imbalance introduce saturation problems for neural networks [211]. The non-i.i.d nature of data sample distribution and class imbalance across worker nodes are practical issues and may arise if the local datasets at each worker node come from mostly (or completely) one particular class.

In that case, that particular local model never sees the samples from the minority class. The proposed method handles these intriguing situations when the data is distributed across the local workers such that, the total data is severely imbalanced (in the case of under-represented class <10%) and some of the local workers may not get any samples of the under-represented class. The present manuscript proposes an alternative by introducing a cost-sensitive momentum averaging scheme. The proposed method has been shown to handle these two problems better than the state-of-the-art methods.

The total size of the actual global dataset is constant. To get severely imbalanced global data, we used some datasets with outliers [73] (one class had <10% of the total samples). The number of samples each worker node receives is chosen carefully by clustering (details in section 6.4) to avoid human bias. Under such conditions, it has been seen that with an increasing number of worker nodes, the performance of the state-of-the-art methods degrades drastically. This degradation is more pronounced when the data is already globally imbalanced. It has been observed that the proposed method is robust in handling the scenario. The chapter, hence, proposes a workaround by introducing the concept of local cost-sensitive momentum averaging. It is seen that for the proposed system, there was minimal degradation in performance and was more robust than the other methods.

6.2 Contributions

In this chapter, it is shown that AE pre-initialisation strategy works best for outlier detection tasks using a neural network when retraining is done by the focal loss function. Further, this chapter proposes an adaptive focal loss for learning from severely imbalanced datasets in a distributed deep learning setting. In this context, a critical class imbalance is explored. We argue that standard methods for federated learning fail to learn from severely imbalanced data and hence face a criticality issue. This is mostly due to either a lesser number of samples at the local worker (which makes the local model overfit) or severely imbalanced partitions at the local worker such that most of the workers do not get a single sample from the minority class. We have shown that as the number of workers is increased, the homogeneity [212] of the local datasets also increase and tends to 1. If all the local datasets contain only a single class sample, then the homogeneity becomes 1. If the local datasets have perfectly identical representations of the classes as the global dataset then the homogeneity becomes zero. For any severely imbalanced dataset, the homogeneity is seen to increase with the number of workers. So, the assumption, that an increasing number of workers result in situations where one or more local datasets have no samples from the minority class (or classes), holds. So, the major contribution of this chapter is as follows:

(i) study on how AE pre-initialisation helps for imbalanced learning (as in outlier detection tasks),

(ii) handling outlier detection in federated deep learning, and

(iii) a novel adaptive focal loss function.

The problem of critical class imbalance in federated deep learning has not been explored or handled in the literature to the best knowledge of the authors. This problem is inherently unique due to its complex nature and resembles a very practical situation (described in section 6.4.1). The basic assumption of i.i.d data distribution across workers is violated in such cases and along with that, the severe class imbalance also affects the performance of the local model for each worker. Related studies have either assumed the data across workers to be i.i.d sampled or having an imbalanced class. However, no method has handled the two issues simultaneously.

There may be a criticism of the assumption that the number of samples in the global dataset is constant and that doesn't make the comparison fair on the grounds of an unequal number of samples for a different number of nodes. In that light, it needs to be explained that the experiment aims to study a practical situation occurs where small chunks of data are generated by several remote devices and data privacy is a concern. The aim was to use the entire training data each time without making any inherent assumption of a local data chunk being more important than the other. This study is different from all the existing studies in this aspect too. The following section describes the related state-of-the-art literature around FL.

6.3 Related Works

The manuscript deals with the concepts of FL for classification using pre-trained fully connected deep neural networks. FL in DL is still in its pristine state. The collaborative learning mechanism is perfect for reaping the benefits of DL in distributed data settings. The era of Big Data and the expertise of DL algorithms have forced researchers to devise novel ways to develop deep distributed learning [199]. The solutions, however, are problem specific. In cases where the data is too big to be contained in a single machine, but the DL model can be trained in a single machine, the data is distributed and the model is replicated across different machines. Each of these machines performs local computations on the models and interacts with each other directly or through a server to understand a global view of the data without transferring the data as a whole. This is called data-distribution architecture [213]. In cases where the data is not so big, but the DL model has so many parameters that it cannot be trained in a single machine, the model is distributed and the data is replicated across different machines. This is called model-distribution architecture [214]. In cases where both the data as well as the DL model are too big for a single machine, both the data as well as the model are distributed. This is known as hybrid-distribution architecture [215]. This manuscript is concerned with data-distribution architecture with the workers and parameter server model shown in Figure 6.1.

In this context, many researchers have come up with methods that are capable of training deep neural networks where data is distributed across worker nodes, also known as Federated Deep Learning [216]. Each of the worker nodes trains a local model on the local dataset that they contain and send the updates (change in weights of the local models) to the parameter server. The parameter server then aggregates the updates or parameters (in this case weights and biases) and sends the aggregated parameters back to the worker nodes for re-initialising the local models after each communication round.

The standard method of aggregation is simple averaging [209], also known as FedAvg. In that case the weights of the parameter server are simple average of the weights of the nodes.

$$w_{param}^{t} = \frac{\sum\limits_{i=1}^{N} w_i^t}{N}, \tag{6.1}$$

where, w_{param}^{t} is weights of the parameter server model at t^{th} round and w_i^t are the weights of the i^{th} worker's local model at t^{th} round. N represents the number of total models participating in the training. This equation is also often written in terms of the weight change vector (Δw) too.

$$\Delta w_{param}^{t} = \frac{\sum\limits_{i=1}^{N} \Delta w_i^t}{N}, \tag{6.2}$$

where, Δw_{param}^{t} is the change in weights of the parameter server model at t^{th} round and Δw_i^t are the change in weights of the i^{th} worker's local model at t^{th} round.

There are several variations on the simple averaging where the sample size each local model gets is also taken into account [206, 209]. In sample size weighted averaging (SSFedAvg), the weights of the local networks are multiplied by the proportion of the total dataset they contain. So, if an i^{th} worker contains d_i samples, then the weight aggregation rule is given as,

$$w^t_{param} = \frac{\sum\limits_{i=1}^{N} d_i w^t_i}{N \sum\limits_{i'=1}^{N} d_{i'}} \tag{6.3}$$

Some other researchers studied the effects of non-i.i.d data settings across the worker nodes and found that the standard variations of FedAvg fail to converge faster. Moreover, the weights of the local models drift apart and the aggregation then requires some adaptive form which prevents the drift. In this context, an adaptive aggregation strategy (FedAdp) was proposed [210] which takes into account the direction of an individual local gradient concerning the global server gradient. The weight update at each i^{th} worker node at communication round t is given by,

$$w^{t+1}_i = w^t_{param} - \eta \Delta F_i(w^t_{param}), \tag{6.4}$$

where, $F_i()$ is the error function at i^{th} worker node and since the local weights are updated by the global aggregated weight at each communication round, the local weight is updated on top of the parameter server weights calculated at the previous round w^{t-1}_{param}. $\Delta F_i(w^t_{param})$ denotes the local gradient of i^{th} worker node's model at communication round t. So, the global gradient is given by,

$$\Delta F(w^t_{param}) = \frac{\sum\limits_{i=1}^{N} d_i \Delta F_i(w^t_{param})}{N \sum\limits_{i'=1}^{N} d_{i'}}. \tag{6.5}$$

The angle of deviation between the global gradient and the local gradient at i^{th} node is given by,

$$\theta^t_i = arc\ cos \frac{\langle \Delta F(w^t_{param}).\Delta F_i(w^t_{param}) \rangle}{||\Delta F(w^t_{param})|| ||\Delta F_i(w^t_{param})||}. \tag{6.6}$$

So, if θ^t_i is small, it means that the local model is updated in the same direction as the global model and would contribute more to the aggregation. The angles are smoothed and a non-linear decreasing function is imposed on it. Smoothed angle in radian is given by,

$$\begin{aligned}\tilde{\theta}_i^t &= \theta_i^t, \quad if\ t = 1 && (6.7)\\ &= \frac{t-1}{t}\tilde{\theta}_i^{t-1} + \frac{1}{t}\theta_i^t, \quad if\ t > 1, && (6.8)\end{aligned}$$

and the non-linear decreasing function is given by,

$$f(\tilde{\theta}_i^t) = \alpha(1 - e^{-e^{-\alpha(\tilde{\theta}_i^t - 1)}}), \tag{6.9}$$

where α is a constant.The adaptive weight for each i^{th} model in aggregation becomes,

$$\phi_i^t = \frac{\sum_{i=1}^{N} d_i e^{f(\tilde{\theta}_i^t)}}{N \sum_{i'=1}^{N} d_{i'} f(\tilde{\theta}_{i'}^t)}. \tag{6.10}$$

So, now the weight aggregation rule becomes,

$$w_{param}^t = \sum_{i=1}^{N} \phi_i w_i^t \tag{6.11}$$

However, even though the method of FedAdp was capable of handling non-i.i.d data settings, it did not take into account cases of extreme imbalance. Moreover, for non-i.i.d data, the number of rounds needed to achieve the desired performance for 10 participating worker nodes was in the order of hundreds.

Another method of aggregation was introduced [217] which could improve the performance of the worker models by allowing them to explore and fluctuate from the parameter's centre variables. They argued that since it is a collaborative learning strategy, there might be numerous local optima. The method is called Elastic Averaging (EASGD). In this method, the aggregated weight is a moving average of the local weights.

$$w^t_{param} = \gamma w^{t-1}_{param} + (1-\gamma)\sum_{i=1}^{N}\frac{w^t_i}{N} \tag{6.12}$$

However, in a situation of local imbalance across the workers for non-i.i.d data setting, it is observed that EASGD fails as the number of worker nodes increases.

A detailed study [218] on some of the methods for selecting some well-learnt workers to update the global model has been done in the context of imbalanced class distribution. However, such methods aim at discarding a number of workers that are supposedly not good enough to contribute to the global model. In doing so, they are rejecting the data that the rejected workers hold and the global model does not get to learn from that particular portion of the data.

Researchers have focussed on the problem of data imbalance across worker nodes and have proposed methods to overcome the challenges [207, 219]. A self-balancing methodology known as Astraea is adapted such that the local datasets in each node are rebalanced by augmentation. However, such methods often work on augmenting the local datasets on workers with the minority class. Thus, have an inherent assumption that the local models have seen at least some actual samples of the minority class. In cases when one or more local models are completely deprived of one or more classes, the problem cannot be handled by minority class data augmentation. Resampling is a popular method to alleviate the imbalance problem by undersampling or over-sampling. However, over-sampling usually is susceptible to errors owing to excess or additional noise, while undersampling lowers the volume of training data from which the model can learn. This makes the above two approaches useless.

Focal loss [220] is a specific loss function that has been presented for a binary classification job that imbalances and reshapes a conventional model cross-entropy loss to weigh down well-classified instances and may concentrate on the learning of the hard imbalanced outliers. The focal loss for binary classification is given as,

$$E = -\rho\{p_1(1-p_2)^{\xi}log(p_2) + p_2(1-p_1)^{\xi}log(p_1)\}, \tag{6.13}$$

where, p_1 and p_2 represent the predicted probabilities of the two classes. Usually for a binary classification problem $p_2 = 1 - p_1$. The constant ρ $(0 < \rho \leq 1)$ is a scaling factor which treats the errors made by the two classes differently and ξ determines how much the easily classified samples (majority class in case of imbalanced data) will contribute to the error. For larger values of ξ, a lesser number of easily classified samples will contribute to error. In a multi-class setting, the focal loss may be written as,

$$E = -\sum_{i=1}^{C} \rho\{p_i(1-\hat{p}_i)^{\xi}log(\hat{p}_i)\}, \tag{6.14}$$

where, C represents the number of classes and p_i and $\hat{p}_i$ represent the predicted probabilities of the class i and not-class i. Focal loss has been directly applied to the federated learning paradigm [204]. However, the method again focused more on selecting well-learnt worker nodes rather than directly handling the imbalance. A similar loss function called GHMC loss [221] (Gradient Harmonizing Mechanism Classification loss) is also proposed to handle outliers in a non-distributed setting. These methods can be straightforwardly applied to federated learning settings as they consider the imbalance of the current worker model. MFE loss [222] is another cost-sensitive method that requires knowledge about the minority classes. It generates a modified form of loss using false positives and false negatives. However, such a piece of prior knowledge is impractical in FL. A new loss function called the Ratio Loss [205] has been proposed recently. The method incorporates a monitoring scheme at each communication round and keeps track of the local imbalance in the training data. However, this loss function again needs to identify the minority class in the global dataset as an added burden. Moreover, it is stated by the authors that as the global imbalance increases, the effective performance of the ratio loss degrades. If the global dataset itself poses an outlier detection problem (where the imbalance is severe i.e. the minority class is only 10% of the dataset), it will fail. Thus, a new loss function is proposed which would modify the local weights in such a manner that even critical imbalance is handled.

Therefore, there is a dire need for an efficient method to handle an extreme class imbalance in local nodes. Moreover, the existing methods of handling class imbalance mostly use pre-trained models. Hence, it has never been studied how the pre-initialisation strategy would affect the learning process in such an imbalanced data setting. Because the training data is consolidated in a centralised setting in typical training situations, there is no differentiation between local and global imbalance values. Thus, minimising the negative impact of imbalance is considerably easier than in the cases of decentralised data. It should be noted that in federated learning, each local model training might be considered as regular centralised training. Intuitively, we might use the existing approaches to handle the local imbalance problem at each round. However, local models exist only momentarily on the local worker nodes and are replaced by the most recent global model after each training cycle. As a result, fixing local imbalances may have little influence. Furthermore, because of the misalignment between local and global imbalances, just implementing known procedures at local devices is often ineffective and may even have a detrimental influence on the global model. Thus, there needs to be a mechanism which can handle each of these local imbalances and the training would work on resolving the global imbalance. The proposed method aims to handle the afore-mentioned issues and also provides a boost in performance. The following section describes the proposed method in detail.

6.4 Proposed Method

The manuscript studies situations where imbalanced data portions may degrade the individual classification ability of the worker nodes. A deep neural network is inherently unsuitable for imbalanced classification due to masking and swamping effects. Thus, researchers have proposed many methods to train a neural network using cost-sensitive backpropagation [220, 221, 222]. As the data is distributed across worker nodes, it becomes relatively difficult to handle the class imbalance. This work does not use any freely available pre-trained models. The local models are first pre-trained using an autoencoder and are then re-trained with class labels. So, each of the local models trains over the data in two phases. Autoencoder pre-training is useful for class imbalance situations because the network gets biased towards learning the majority class' features efficiently. So, any data that doesn't fit into the description of the majority class may be picked up as an outlier easily [16, 73]. The re-training is just done to learn the classification between the majority and minority classes. The following section describes the proposed method to deal with the severe class imbalance in a distributed setting.

6.4.1 Division of Data Across Worker Nodes

The data is arbitrarily distributed and resembles a practical scenario where a user has no control over the number of samples each node gets. In both cases, the datasets were partitioned and given to each worker node. Partitioning the data into non-overlapping portions makes the data across local workers 'non-independent'. To remove the 'identically-distributed' assumption of i.i.d, the global data clustered and each cluster represented a partition of the dataset. For this, we used k-means clustering where k is equal to the number of worker nodes. Thus, it was ensured that the local datasets that the worker nodes get are non-i.i.d. As the number of clusters grows (which is the same as increasing the number of local workers), the probability of each cluster having samples from only a single class increases [212]. Thus, as the number of local workers grows, there is an increased chance that the local datasets are mostly of a single class only. This method of data division is merely mimicking a practical data distribution scenario across the workers which assumes that the data generated by each local worker is mostly similar.

To understand the practical implication of this assumption, let us consider an example where there are two branches of a company located across two different locations in the world. It may happen that for one of the locations, the workers are mostly relatively younger and for the other location the workers are mostly older. Thus, the first company can produce goods at a much faster rate with fewer errors while the second company produces goods at a slower pace and with more errors than the first one. Thus, the data obtained from the first location will have more samples and a majority of the samples will pass the quality check. On the other hand, the data from other locations will have lesser samples and a majority of the samples will not pass the quality check. So, if one attempts to learn the model of a quality check from both companies, he or she needs to learn in an imbalanced and distributed setting simultaneously.

6.4.2 Cost-sensitive Neural Network Training

The focal loss may be extended to a cost-sensitive federated learning scenario by incorporating an adaptive scaling factor in each local model. Since each model gets a different number of samples and the class distribution in each model is different, we need to adapt the value to ρ accordingly. So, an adaptive focal loss function is proposed in this context. For a C-class classification problem involving N models (one in each local worker node), each local model j would have the proposed focal loss as,

$$E_j = -\sum_{i=1}^{C} \rho_j \{p_i(1-\hat{p}_i)^{\xi} log(\hat{p}_i)\}, \tag{6.15}$$

where, C represents the number of classes, p_i and $\hat{p}_i$ represent the predicted probabilities of the class i and not-class i and ρ_j is the adaptive scaling factor of the j^{th}, local model. Since, the main motive for imbalanced data is to avoid the masking phenomena (labelling all the samples with the majority class label), it is intuitive to express ρ_j in terms of an crude class imbalance ratio m_j for each local dataset, such that,

$$\rho_j = \frac{a}{(1 + e^{-b*(m_j - 1)})} \tag{6.16}$$

where,

$$m_j = 1 - \frac{d_j^{majority}}{d_j^{total}}. \tag{6.17}$$

Here, $d_j^{majority}$ and d_j^{total} represent the number of samples in the majority class of the local dataset in j^{th} worker and the total number of samples in the local dataset in j^{th} worker respectively. A crude class imbalance is much simpler than having class wise imbalance ratio for extremely imbalanced datasets. This is because it is only the majority class which overpowers the minority classes. The other minority classes do not affect each other much as they are themselves smaller in number. For the present work, the values of a and b were chosen to be 2 and 3 respectively. So, if a local dataset has all the samples from the majority class only ($m_j = 0$), the local model would be completely masked and must not be able to make decent contributions to the error. So, if $m_j = 0$ then instead of pushing ρ_j to zero, we are just mapping it to a value close to zero (but not exactly zero) using the scaled sigmoid function. Pushing ρ_j to zero would mean that the model is correct and will not add anything to the server weights. This would imply a complete loss of information from one worker which is undesirable. Thus, the value of ρ_j being close to zero would mean that the local models would at least have some small contribution to the server model.

Another intuitive explanation for this adaptive function is that each model is independently moving towards convergence. This rate of movement depends on the individual loss they incur. Now, if a local model is overwhelmed with severe imbalance, then the constant ρ_j would prevent that model to overfit the majority class too much. This would imply that the more severe the imbalance a local model would face, the lesser it would deviate from the last global update, but the information would not be completely lost. This would help the global weights not drift randomly. It can be proved that there would always at least be one local data portion that will not have the severity in imbalance, and the model trained on that particular data would always govern the global weights. However, that model alone is not enough to learn the entire data space. So, the other models which are inflicted with severe imbalance would add the fine-tuning to the global weights accordingly.

Theorem 1. *If the global dataset P has an outlier ratio r, and it is partitioned across N divisions namely, P_1, P_2... P_N, then there would be atleast one division P_i which would have outlier ratio $\geq r$.*

Proof. Suppose the global data having n_1 samples from the minority or outlier class (denoted as class 1) and n_2 samples from the majority or inlier class (denoted as class 2), is partitioned into N local datasets. The global outlier ratio r is given by,

$$r = \frac{n_1}{n_1 + n_2} \tag{6.18}$$

For the i^{th} partition (local dataset) P_i, let there are $n_{i,1}$ samples from the minority class and $n_{i,2}$ samples from the majority class. Then the outlier ratio of that partition is given by,

$$r_i = \frac{n_{i,1}}{n_{i,1} + n_{i,2}} \tag{6.19}$$

Now, as the global dataset is partitioned into N local data partitions,

$$\sum_{i=1}^{N} n_{i,1} = n_1, \ and \tag{6.20}$$

$$\sum_{i=1}^{N} n_{i,2} = n_2. \tag{6.21}$$

Equation 6.19 can be written as,

$$n_{i,2} = (1 - r_i)n_{i,1}, \tag{6.22}$$

and, from equation 6.18, we get

$$n_2 = (1 - r)n_1. \tag{6.23}$$

Replacing equations 6.20 and 6.21 in equation 6.23, we get,

$$r = \frac{\sum\limits_{i=1}^{N} r_i n_{i,1}}{\sum\limits_{i=1}^{N} n_{i,1}} \tag{6.24}$$

For argument, if we assume that for all partitions, the outlier ratio r_i is strictly less than the the global outlier ratio r, then,

$$r_i < r,\ for\ i =\ 1,\ 2,\ 3,\ ...,\ N. \tag{6.25}$$

Multiplying both sides of equation 6.25 by $m_{i,1}$, summing over all the partitions on both sides and rearranging we get,

$$r > \frac{\sum\limits_{i=1}^{N} r_i n_{i,1}}{\sum\limits_{i=1}^{N} n_{i,1}} \tag{6.26}$$

It can be seen that the equations 6.24 and 6.26 contradict each other. Thus, there exist atleast one partition of local data which has an outlier ratio $\geq r$. $\square$

6.4.3 Overall Method

EASGD has been proven to be stable [217] in cases of distributed deep learning. It induces an elastic force that does not allow the local model's weight matrix to deviate too much from the global weight matrix whereas also allowing some independence. The proposed method incorporates the concept of EASGD with the proposed adaptive focal loss [equation 6.15]. The training of the network is done in two phases.

(i) pre-training the network using deep autoencoders and

(ii) retraining the entire network using class labels.

While pre-training the network, since there are no class labels involved, and the models merely learn to reconstruct the input, the unnecessary involvement of the adaptive cost-sensitive loss function is not needed. So, the autoencoders are trained by the standard EASGD (equation 6.12) to minimise the reconstructional mean squared error loss. Now, for the retraining, the model must learn the discrimination between classes even in presence of severe class imbalance. So, retraining needs the cost-sensitive loss function along with EASGD. The cost-sensitive adaptive focal loss function determines how much independence the individual local model would get in determining the optimum weights depending on the class imbalance they witness.

The pre-training without cost-sensitivity allows the autoencoders to distinguish between the majority and the minority classes. It has been proven that autoencoders map similar values to similar ranges [16] and thus for classification in such severe imbalanced datasets, they are very useful. Autoencoders are extensively employed in outlier identification [54, 223], where an input with a substantial reconstruction error is identified as out-of-distribution, as the quality of reconstruction is expected to decrease for inputs that are considerably different from the majority of the training data. The traditional AE-based pre-training is also favoured since it is highly stable and simple, as well as pro-

Table 6.1: Dataset description

Name of the dataset	Number of Samples	Number of features	Percentage of Outliers
Credit Card	284807	29	0.1761
HTRU2	17898	8	9.11
MNIST	7603	100	8.7373
Pulsar	17898	8	9.204119
Forest Cover	286048	10	0.9484
Mammogram	7828	6	2.325373
Pen Digits	4808	16	2.271404
ALOI	34999	28	3.016121

vides a decent initialisation, to begin with. So, the retraining is to just fine-tune the local models according to the proportion of the different classes present in the local dataset.

6.5 Experiments and Results

To establish the supremacy of the proposed model, the study explores many different angles on an empirical note. For a fair comparison between the methods, the number of models was increased. The model architecture was kept the same for all the methods. All the methods were compared with the same partitions of the data produced by k-means clustering (as assumed in section 6.4.1). Each of the methods was run for only 20 communication rounds between the parameter server and the workers. The datasets used were highly imbalanced. We have chosen eight binary-class outlier detection datasets for this purpose which had the outlier class samples representing less than 10% of the total dataset. The description of the datasets is provided in Table 6.1. Table 6.2 provides the description of the architecture of the autoencoder used for various datasets.

The following sections provide the results of the experiments conducted. The results reported are the median over ten independent runs.

Table 6.2: Architecture of the autoencoder used for various datasets.

	Number of layers	**Number of nodes in each layer**
Credit Card	3	10
HTRU2	16	7
MNIST	4	20
Pulsar	3	4
Forest Cover	9	4
Mammogram	3	4
Pen Digits	9	5
ALOI	3	10

6.5.1 Autoencoders versus other Feature Extraction Methods

Since a standard neural network fails to detect outliers and is often biased towards the majority class, we want to see the performance with the focal loss which is designed for imbalanced datasets. In doing so, we also need to examine the usefulness of an AE. So, we replaced the layers of the network which are pre-trained with AE by different feature extraction methods like k-PCA, t-SNE, LLE, Isomap and GRP and used the same neural network's classification layers for all. The classifier network minimises focal loss function in all the cases. Since the other feature extraction methods are not undergoing any re-training, we have frozen the weights of the AE layers to have a fair comparison.

Table 6.3: ROC-AUC Obtained by Various Feature Extraction on the Imbalanced Classification Performance

	AE	**kPCA**	**tSNE**	**LLE**	**Isomap**	**GRP**
Credit Card	**0.9197**	0.5	0.5	0.5	0.5992	0.8084
HTRU2	**0.9301**	0.5	0.8094	0.7505	0.8762	0.8987
MNIST	**0.9423**	0.5262	0.7282	0.7364	0.5	0.9207
Pulsar	**0.9387**	0.5	0.8727	0.7628	0.8795	0.8777
Forest Cover	**0.9798**	0.5	0.5	0.5	0.5	0.5
Mammogram	**0.8525**	0.5	0.7545	0.5	0.5201	0.8492
Pen Digits	**0.9543**	0.6292	0.9212	0.5	0.9456	0.9477
ALOI	**0.9612**	0.5	0.6618	0.7048	0.5	0.5

It can be seen from Table 6.3 that the model using AE is the best for all datasets. There are some methods which are comparable in performance to AE in some datasets, but they are failing for others. Moreover, as AE can be fine-tuned, it is much more convenient to use AE and increase the classification performance.

6.5.2 How Pre-initialisation affects Outlier Detection?

The initialization phase is essential to the model's eventual output and mandates an effective approach. Traditionally, weight initialization entailed employing smaller random values (close to zero). However, over the last decade, better explicit heuristics that utilise network architecture information, for example, the type of activation being utilised or the number of inputs to the network, have been developed. These rather specific strategies can lead to more successful neural network model construction when using the stochastic gradient descent optimization algorithm. This optimization procedure necessitates a starting point in the universe of possible weight values wherein the optimization may commence. Deep model training is a challenging enough process that the initialization method used has a significant impact on most techniques. The starting point can decide whether or not the process converges at all, with certain initial values being so unstable that the algorithm runs into numerical challenges and fails. Many initialisation strategies are gradually developed. Some popular methods used these days are Random Normal [224], Random Uniform [224], Xavier-Glorot Normal [5], Xavier-Glorot Uniform [5], He Normal [225] and He Uniform [225]. AEs were also proposed as a pre-initialisation strategy to avoid the problem of vanishing and exploding gradients [15]. While techniques like AE pre-training certainly laid the groundwork for many essential principles in today's deep learning methodologies, the compelling necessity for pre-training neural networks has waned in recent times. This was largely due to various advancements in regularisation, network architectures, and enhanced optimization methods. Despite these advancements, training deep neural networks to generalise well to a wide range of complicated tasks remains a major issue. One of these is the issue of imbalanced datasets. Neural networks get biased towards the majority class. However, as observed in chapter 3, AEs are well suited for outlier detection tasks. Hence, to strengthen the position of AE pre-training in outlier detection tasks, it was compared with the popular initialisation strategies. The models use various initialisation schemes and the training for classification is done using the focal loss function [equation 6.13] (because for a single model, the equations 6.13 and 6.15 are identical). Table 6.4 gives a comparison of how the various initialisations perform for severely imbalanced datasets (where the minority class has less than 10% representation) with respect to all the four metrics namely ROC-AUC (Area Under the Curve of Receiver Operating Characteristic) [73], F-Score [105], DR (Detection Rate) 3.12 and G-Mean [105]. It can be seen that AE pre-training added advantage to the performance of a deep neural network for all the eight datasets under consideration.

It can be seen that AE pre-trained models are better in performance for outlier detection tasks. This supports the findings as observed in chapter 3 that AE are better feature extractors for outlier detection tasks. Thus, the experiments can now be extended to multiple models in a federated setting as described in the next sections.

6.5.3 Data Division and Critical Imbalance

Data division in a non-i.i.d setting violates two main properties of i.i.d [208]. Firstly, the data is divided in an insufficiently random order, which generates a violation of independence. Secondly, there are varying amounts of data in each division (concerning class labels or sample quantity, or both). This implies that each data division must follow a separate distribution which is not identical to the global data distribution. Let the global data distribution is $\mathcal{D}$, then in an i.i.d data division, every i^{th} portion of data $\mathcal{P}^{(i)}$ has the same distribution $\mathcal{D}$. However, for non-i.i.d data division, every i^{th} portion of data $\mathcal{P}^{(i)}$ might follow a different distribution $\mathcal{D}^{(i)}$, which means that there is an underlying structure in each data portion. To mimic this property, one may safely consider clustering the data into k groups. In this work, the clustering has been done using k-means clustering algorithm. This preserves the non-i.i.d properties that the

Table 6.4: Effect of Various Network Initialisations on the Imbalanced Classification Performance

Dataset	**Metrics**	**AE-Pretraining**	**Random Normal**	**Random Uniform**	**Xavier-Glorot Normal**	**Xavier-Glorot Uniform**	**He Normal**	**He Uniform**
Credit Card	**ROC**	**0.9537**	0.5000	0.9339	0.9365	0.9379	0.9323	0.9461
	F-Score	**0.9346**	0.5000	0.9036	0.9159	0.9176	0.9098	0.9104
	DR	**0.8083**	0.0000	0.7870	0.7799	0.7799	0.7870	0.7941
	G-Mean	**0.9351**	0.5000	0.9037	0.9163	0.9180	0.9100	0.9105
HTRU2	**ROC**	**0.9756**	0.5000	0.9726	0.9737	0.9736	0.9737	0.9731
	F-Score	**0.9425**	0.0000	0.9309	0.9340	0.9348	0.9336	0.9316
	DR	**0.8491**	0.0000	0.8450	0.8343	0.8407	0.8364	0.8257
	G-Mean	**0.9426**	0.0000	0.9309	0.9342	0.9349	0.9337	0.9317
MNIST	**ROC**	**0.9952**	0.9831	0.8459	0.9907	0.9885	0.9856	0.9885
	F-Score	**0.9548**	0.9359	0.7506	0.9340	0.9422	0.9278	0.9355
	DR	**0.9339**	0.8734	0.3696	0.8550	0.8920	0.8595	0.8642
	G-Mean	**0.9549**	0.9359	0.7537	0.9340	0.9422	0.9278	0.9355
Pulsar	**ROC**	**0.9797**	0.5000	0.9752	0.9733	0.9693	0.9723	0.9758
	F-Score	**0.9528**	0.0000	0.9400	0.9449	0.9250	0.9325	0.9387
	DR	**0.8590**	0.0000	0.8153	0.8369	0.7669	0.7909	0.8150
	G-Mean	**0.9530**	0.0000	0.9405	0.9452	0.9257	0.9330	0.9391
Forest Cover	**ROC**	**0.9999**	0.5000	0.9994	0.9989	0.9993	0.9994	0.9995
	F-Score	**0.9910**	0.0000	0.9805	0.9779	0.9794	0.9778	0.9777
	DR	**0.9774**	0.0000	0.9524	0.9310	0.9690	0.9476	0.9548
	G-Mean	**0.9910**	0.0000	0.9805	0.9780	0.9794	0.9778	0.9777
Mammogram	**ROC**	**0.9389**	0.5000	0.9223	0.9306	0.9311	0.9341	0.9231
	F-Score	**0.8817**	0.0000	0.8565	0.8362	0.8365	0.8413	0.7470
	DR	**0.6980**	0.0000	0.6844	0.5892	0.6163	0.5756	0.3731
	G-Mean	**0.8823**	0.0000	0.8566	0.8373	0.8370	0.8431	0.7499
Pendigits	**ROC**	**0.9876**	0.5000	0.5000	0.9706	0.9779	0.9753	0.9761
	F-Score	**0.9660**	0.0000	0.0000	0.9339	0.9486	0.9601	0.9495
	DR	**0.9763**	0.4888	0.4888	0.9049	0.9636	0.9755	0.9543
	G-Mean	**0.9112**	0.0000	0.0000	0.9334	0.8670	0.8891	0.8891
ALOI	**ROC**	**1.0000**	0.5000	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	F-Score	**0.9994**	0.0000	0.9966	0.9983	0.9983	0.9977	0.9977
	DR	**1.0000**	0.0000	0.9908	0.9954	0.9954	0.9931	0.9931
	G-Mean	**0.9994**	0.0000	0.9966	0.9983	0.9983	0.9977	0.9977

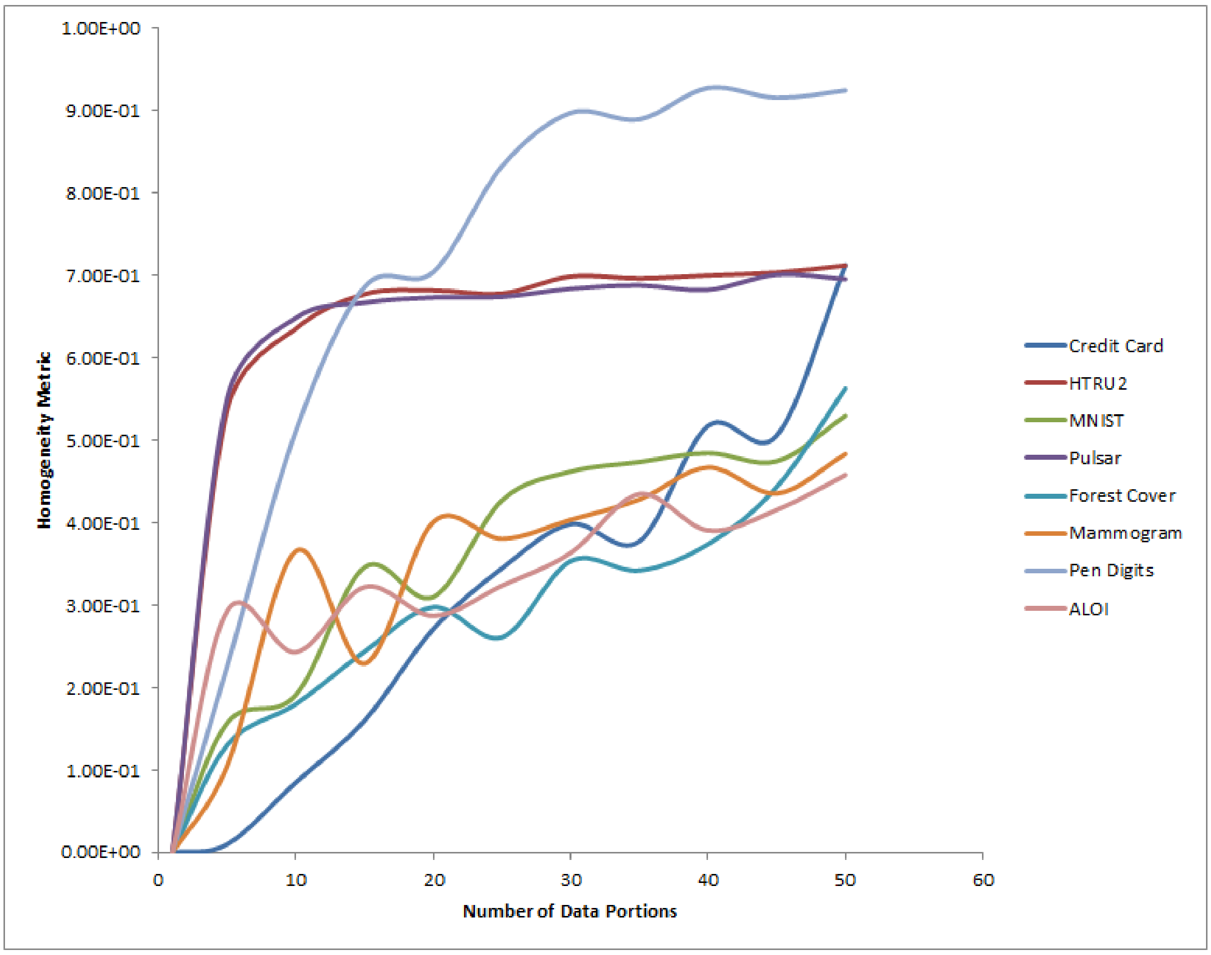


Figure 6.2: Variation of the Homogeneity Metric with the Number of Data Portions

data portions do not follow an identical distribution (as each cluster consists of similar samples and samples from different clusters are dissimilar to each other) and are also not independent (as both the proportion of class labels and the number of samples vary in each cluster). The number k is not always equal to the number of classes in the dataset. This is because any clustering algorithm groups data according to the similarity of the samples and not according to the labels. Homogeneity metric [212] indicates the conditional entropy of a clustering algorithm in identifying the correct classification and is bounded between 0 to 1. This means that if more clusters contain samples from only one class, the score will increase. This is evident from our findings too [Figure 6.2]. It can be seen that as the number of data portions increased (or number of worker nodes increased), there were multiple clusters which had no samples from the minority class

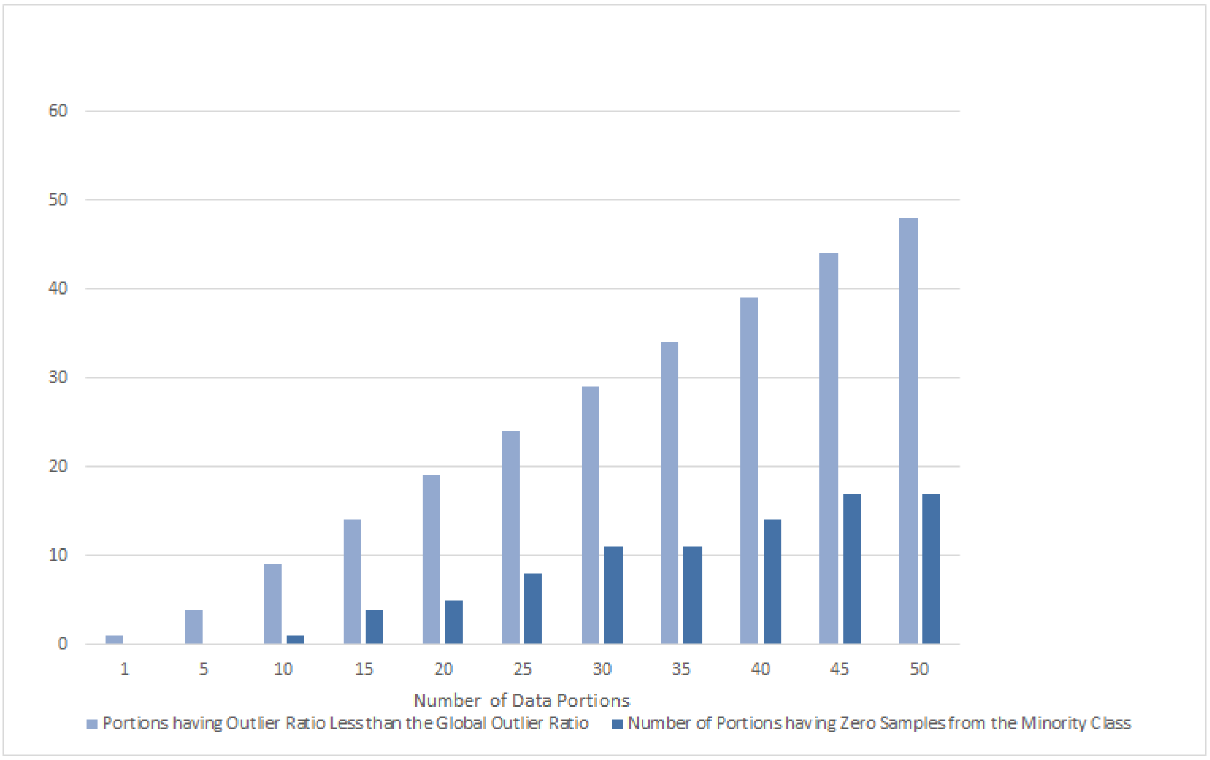


Figure 6.3: Imbalance Becomes Critical with the Increasing Number of Data Portions

[Figure 6.3] (result shown for Credit Card dataset). The number of data portions having an imbalance ratio below the global imbalance ratio kept on increasing with the number of data portions (or the number of worker nodes) [Figure 6.3]. This indicates a critical imbalance situation that federated learning suffers from.

6.5.4 Performance under Critical Imbalance

As can be seen from Figure 6.3, with an increasing number of partitions, there are multiple local datasets which suffered critical imbalance (i.e. have no samples from the outlier class). Under such conditions, we need to analyse how the performance of the global model varies with each communication round. For experimental purposes, we have taken 10 worker nodes for the Credit Card dataset. It can be seen from Figure 6.3 that for 10 worker nodes, there was 1 data portion which had zero samples from the outlier class and 9 data portions having an outlier ratio less than the global outlier ratio (including the one which has no outlier samples). Figure 6.4 shows the F-Score obtained on the

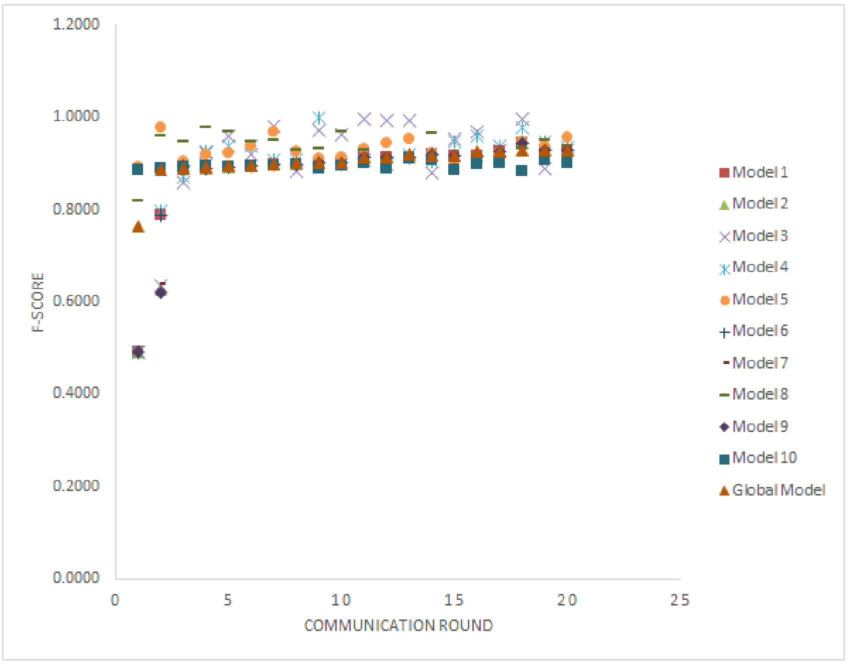


Figure 6.4: F-Score of Individual Local Models and the Global Model vs Communication Round

test data for each local model before each communication round and the global model after each communication round. This is a clear indication of the stability of the elastic averaging [217].

It can be seen from figure 6.4 that the individual models gradually coincided with similar performance on the test data with each communication round. It can be seen that several models were not performing well in the initial rounds. Models 1, 2, 3, 4, 6, 7 and 9 were only capable of detecting the samples from the majority class after the first round. However, due to the learning obtained from the other well-performing models, they gradually shifted to a better performance in classifying the samples. This indicates that even though there are some underperforming models in the learner pool, the global model will not be so severely affected by it and there will be gradual learning eventually. However, the maximum number of under-performing models allowed for the global model to learn properly has not been experimented with in this thesis. This can be a future direction of research in this regard.

6.5.5 Number of workers vs overall performance

Under the above conditions, another important aspect is to study how the increasing number of worker nodes affects the overall performance of the global model. In the case presented, the total data is always constant and each worker gets a unique partition of the dataset. Thus, as the number of worker nodes increases, the number of samples in each worker node will decrease. So, increasing the number of workers above a certain limit is not possible to prevent the unlikely condition of models getting insufficient samples for training. It has been seen that all the existing methods tend to be biased towards the majority class as the number of workers grows. It is noteworthy that by increasing the number of workers in the present assumption of experiments that the data samples are partitioned (i.e. increasing the number of workers would decrease the number of samples in each partition), not having enough quantity of the data in each worker node will be an inevitable cause of performance degradation. Hence, the study mainly shows that the proposed method yields a consistent performance with increasing worker nodes while the existing methods hit a saturation point after a certain limit. The proposed method has been compared with the existing methods used for outlier detection in federated learning: EASGD (with cross-entropy loss) [217], Astraea[219], FedFocal [204], and Ratio Loss [205].

As observed from the above experiments (Figures 6.5-6.12), the outlier detection abilities of EASGD (with cross-entropy loss) suffer a fast degradation with an increasing number of nodes. Astraea improved on the detection abilities but with an increasing number of nodes, showed incompetence. The Ratio Loss method and the method FedFocal were more or less similar in performance and were able to detect outliers fairly up to an appreciable number of nodes before suffering failure. The proposed method had the best performance among all the methods and the performance was more or less steady with an increasing number of nodes and was still able to detect outliers while all the

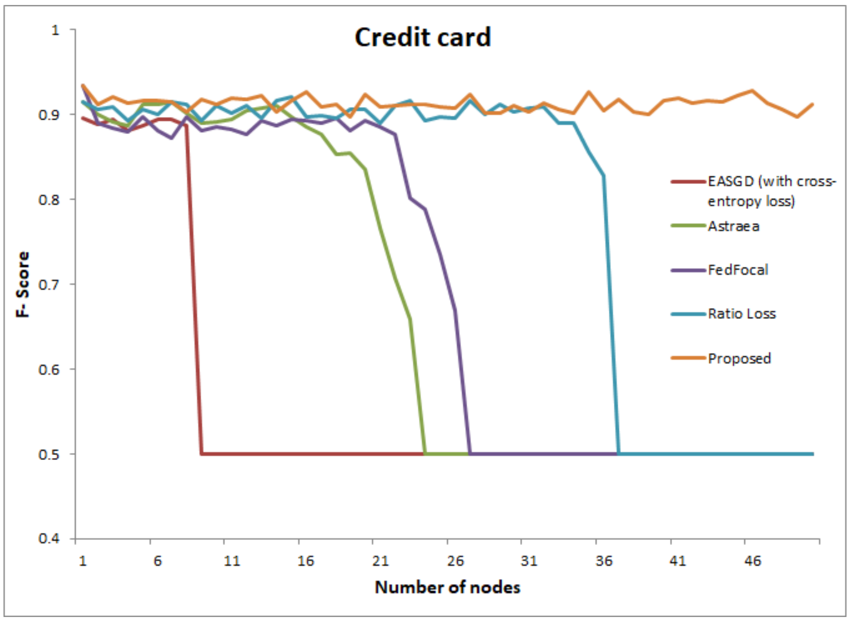


Figure 6.5: Variation of F-Score Obtained by Various Methods for Different Numbers of Worker Nodes on Credit Card Dataset.

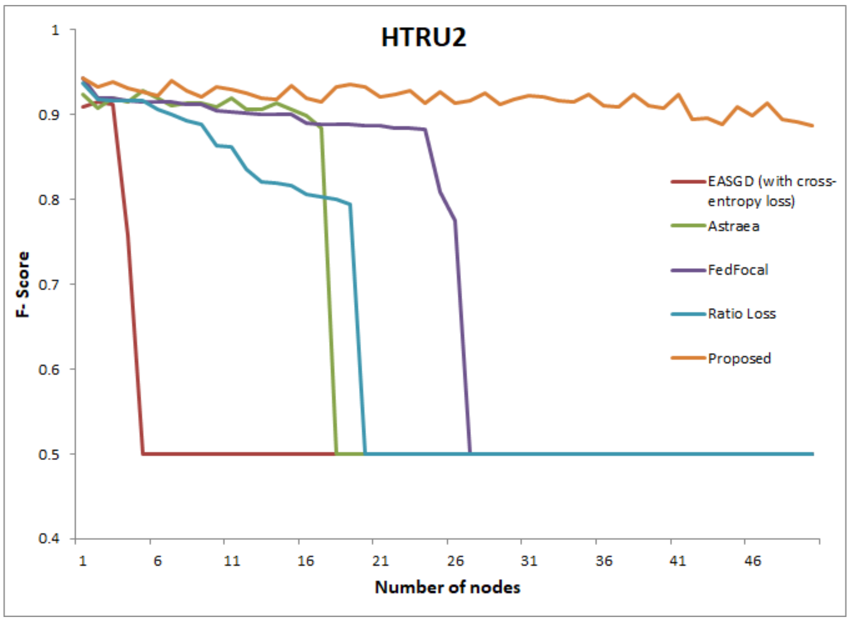


Figure 6.6: Variation of F-Score Obtained by Various Methods for Different Numbers of Worker Nodes on HTRU2 Dataset.

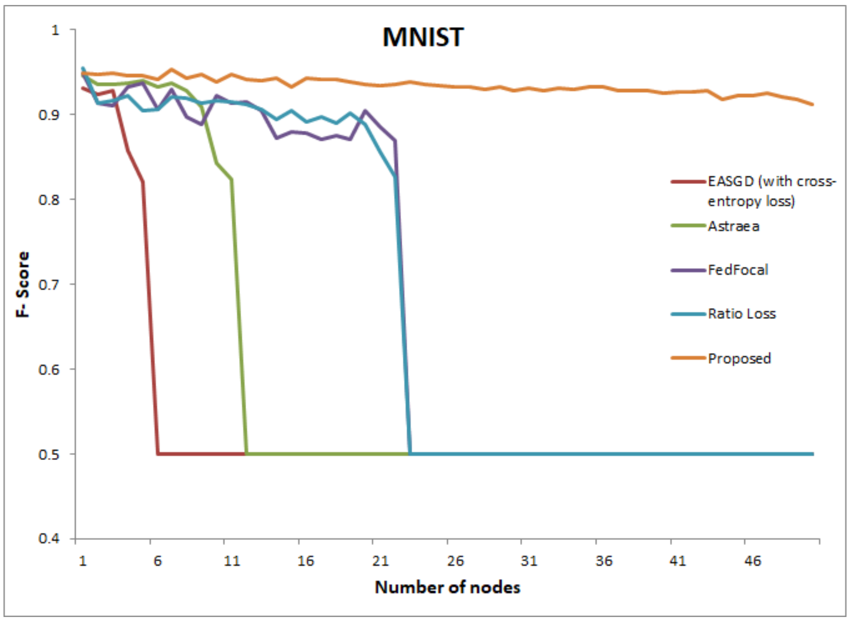


Figure 6.7: Variation of F-Score Obtained by Various Methods for Different Numbers of Worker Nodes on MNIST Dataset.

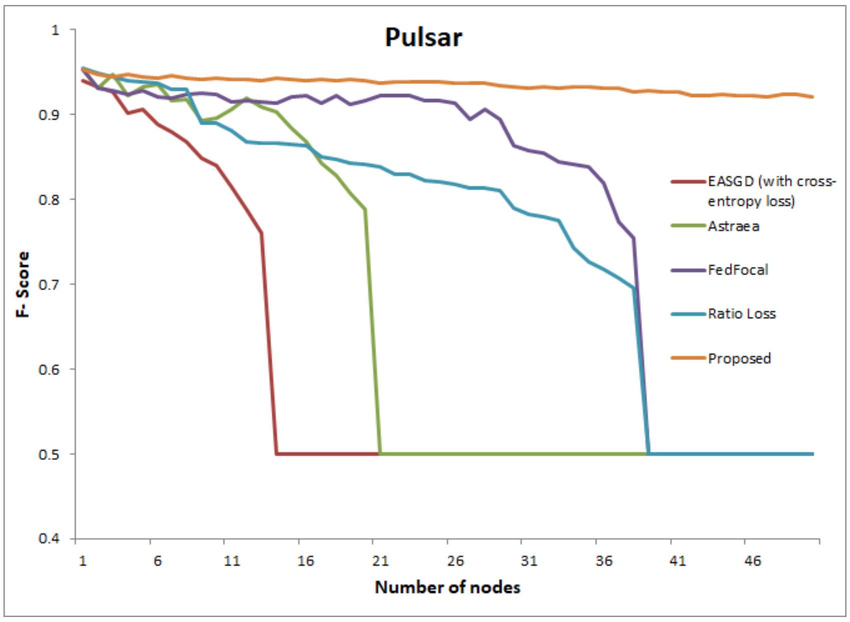


Figure 6.8: Variation of F-Score Obtained by Various Methods for Different Numbers of Worker Nodes on Pulsar Dataset.

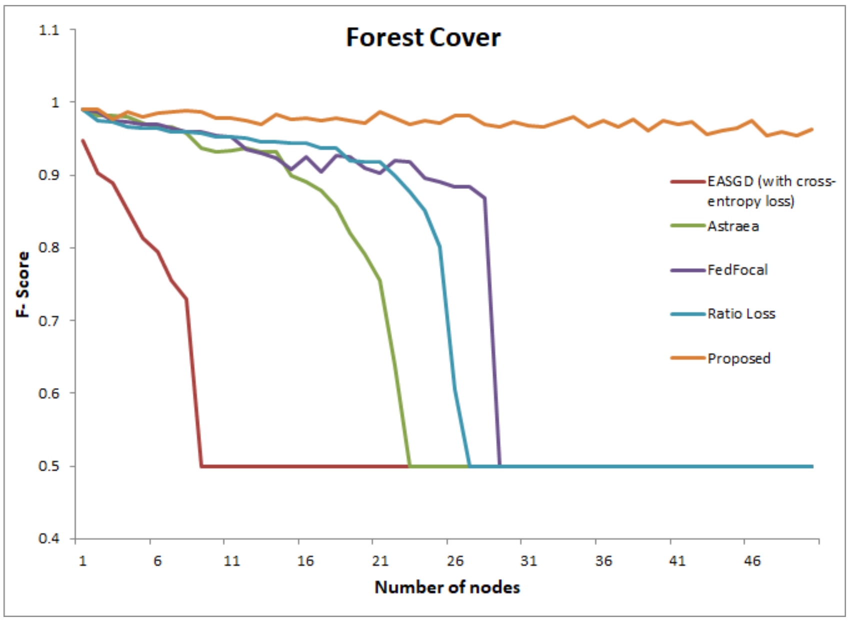


Figure 6.9: Variation of F-Score Obtained by Various Methods for Different Numbers of Worker Nodes on Forest Cover Dataset.

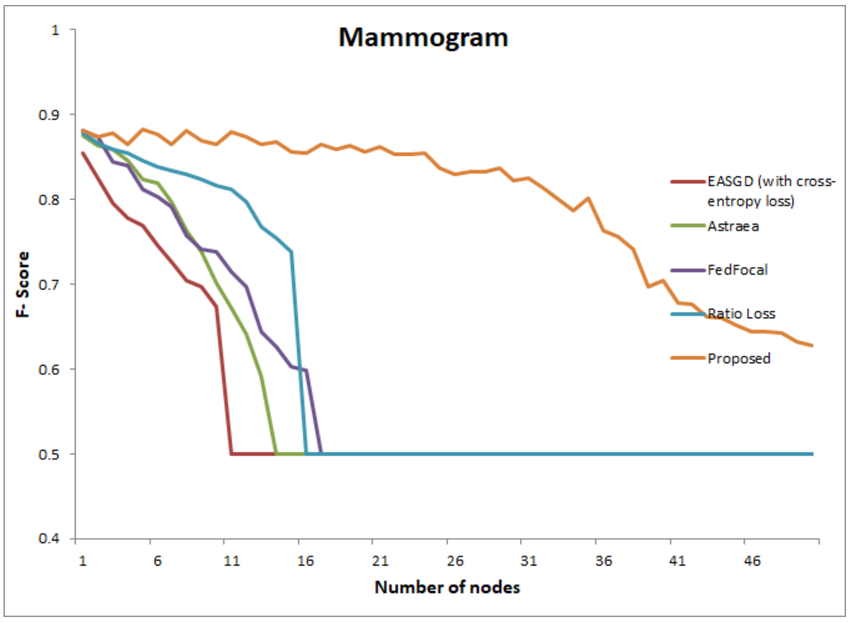


Figure 6.10: Variation of F-Score Obtained by Various Methods for Different Numbers of Worker Nodes on Mammogram Dataset.

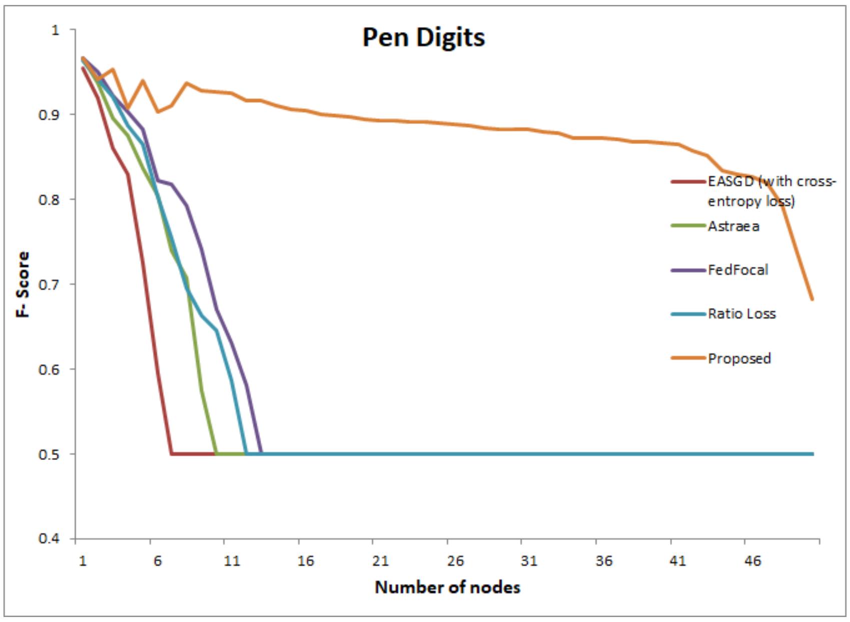


Figure 6.11: Variation of F-Score Obtained by Various Methods for Different Numbers of Worker Nodes on Pen Digits Dataset.

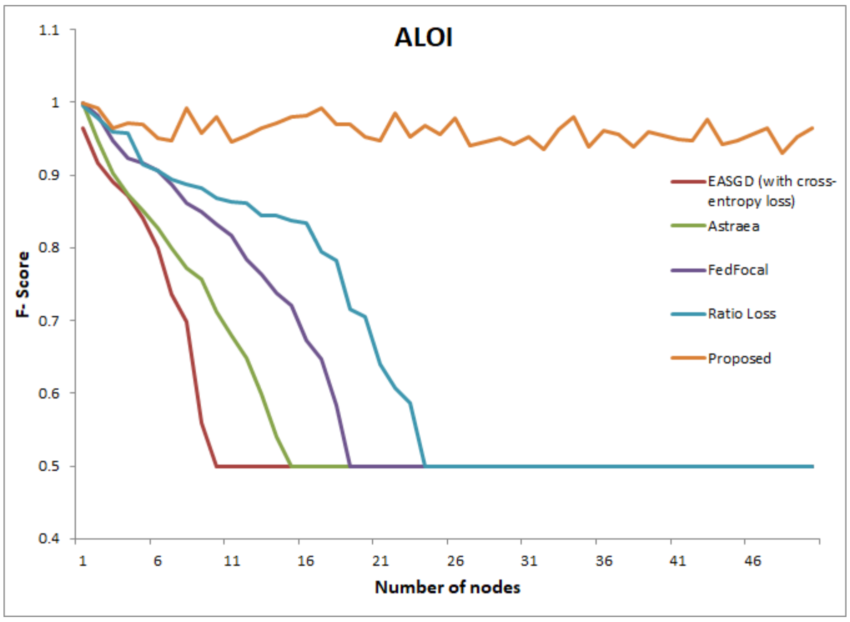


Figure 6.12: Variation of F-Score Obtained by Various Methods for Different Numbers of Worker Nodes on ALOI Dataset.

detection abilities of the other existing methods already collapsed. There was a gradual decrease in the performance of the proposed method for some datasets due to the lack of enough training samples in each local partition with the increasing number of nodes. This is an expected consequence of our assumption that the total number of samples in the global dataset is constant.

6.6 Discussion and Conclusion

In this chapter, it has been shown that AE pre-training is better for outlier detection tasks. The study is then extended to the use of AE pre-training in a distributed setting. In this context, an adaptive version of the focal loss function is introduced to handle the critical imbalance in the distributed data sets. The main assumptions in the experiments were that the data available at each local node is mutually exclusive. If one local node fails, it is assumed that there is a backup node for each local node which would continue with the learning process. Under such assumptions, it has been observed that the proposed method can handle the critical imbalance better than the existing methods. The experiments show that even when some local datasets did not have any samples from the outlier class, the proposed method drove the global model and each of the local models to similar performance on the test data.

However, there might arise an extreme case of imbalance where one local model may contain a lot of outlier samples such that the outlier class becomes the majority for that particular local dataset and all the other local datasets do not contain any outlier sample at all. In that case, the present solution might not work as it relies on the outlier ratio. Such an extreme case would practically imply that all the anomalies are generated at the same location while others do not generate any at all. This problem needs to be handled in a different light and has been set aside as a direction for future research. The research may also be extended to handle multiple classes where the local imbalance is critical and the global imbalance may or may not be present.

Chapter 7

Conclusion, Discussion and Scope of Further Work

7.1 Conclusion

AEs usually follow a bottleneck design which is believed to be the key feature for detecting efficient representations of the data. To avoid the bottleneck one might also include sparsity or introduce skip connections. The main task of an AE is to reconstruct the input. However, the output is not a replica of the input and just shares a good amount of similarity. Thus, the features extracted in the hidden layers are very interesting to notice. Unlike PCA (where the projections are orthogonal), AEs bring out some really interesting correlated projections too. The intuition is that the hidden layer must need to capture the essential amount of information which is sufficient to reconstruct the input at the output layer. However, just like an NN, AEs learn what they see. Thus, the patterns which are frequently seen by the AEs are often learnt much better than the rare patterns. This is a particularly peculiar observation which makes an AE suitable for tasks where repetitive samples overwhelm a model. This has been the main idea behind the tasks that the thesis aimed to handle. It is observed that using AE as a supplement boosted the performance significantly.

The thesis establishes AE as a beneficial addition to the models. Other existing feature extraction methods might give comparable results in some specific datasets but are not as versatile as AE. Still, the relevance of features extracted by an AE can only be fine-tuned through supervision or can be selected by some particular criteria fit for the task. It should also be mentioned that although pre-training by AEs might not be beneficial against other initialization strategies when data is imbalanced AE pre-training works slightly better. This has been shown in the different chapters of the thesis. However, there are also some aspects which are unexplored and overlooked in each chapter.

7.1.1 Chapter 3

Most of the datasets existing are not a hundred per cent perfect. An individual will be incredibly lucky if they ever receive a perfectly balanced dataset. Usually, the datasets are such that each class has a varying number of samples. Concern for the minority classes is only really important if we worry about them. Suppose we are attempting to determine whether or not we should buy a property in light of the market's condition, its features, and our budget. This is a significant investment, so it is vital to make the correct selection. If the model says not to buy the property when it should be bought actually, then the buyer does not lose anything. There are other properties available which he may choose from. However, making a poor investment in such an important asset would be disastrous. When it decides on buying, the class that represents the "buy" signal is critical and the decision must be correct. The class that represents the "not-buy" signal is not so important. However, in practice, since purchasing a property is considerably more unusual than not buying, the model will be biased towards learning the "not-buy" class extremely well since it constitutes the maximum percentage in data. The model could do badly in the "buy" class since it has fewer representative data points.

Now, if we aim to achieve the highest accuracy without giving any undue importance to any of the classes, then we should not think much about detecting the minority classes. Most of our accuracy will be due to the classes constituting most of the training examples. Classification problems with an unequal number of instances per class are referred to as imbalanced classification problems. However, it must not be confused with the 'unbalanced' classification problem. Unbalanced cases occur when the original data was balanced, but due to some reason, it has lost the balance. An example may be the scenario of the ratio between male and female babies born at a particular place. The situation is more or less balanced technically. If it is not balanced, then there is a definite and peculiar reason behind it. However, imbalanced classes are inherently skewed and represent rare occurrences. For example, if we consider normal healthy babies and babies with genetic anomalies, the number of normal babies is higher than babies with genetic anomalies. There is a definitive physics working behind it. No matter how much we try, we cannot alter this imbalance in any way.

Although ML research tries to handle both problems unanimously, the two different problems need different outlooks. This may be a newer direction of research which can be pursued. As per my belief, the methods which include upsampling or downsampling of the datasets are conceptually much more suited for unbalanced classification than imbalanced classification tasks. This is because sampling forces the data to look balanced. If the dataset's core property is in its skewed nature, then it is not appropriate to use sampling and destroy the core property.

The area of unbalanced datasets is still not recognised as a separate direction of research. Moreover, the issues of imbalanced classification also need to be explored given stronger algorithms which are more robust to handling the skewness directly. Most of the algorithms work well with perfectly separable classes irrespective of whether they are balanced or not. So, efficient feature representations are necessary which would enhance the separability between the classes.

In this thesis, AEs are explored as a solution to increase the class separability in severely imbalanced datasets which contain outliers or anomalies. In future, AEs may also be used to solve the problems of unbalanced classification by generating samples of the minority classes using generative models like generative adversarial networks or some other modified models. Whether it be imbalanced or unbalanced classification, they come with their own set of issues. Both the problems are unique and challenging in the fact that they reveal a new physics behind the occurrences. The former reveals why a particular 'normal' behaviour is not seen in some samples and the latter reveals why the entire dataset is behaving 'abnormally'.

7.1.2 Chapter 4

There are concerns about samples changing their behaviour over a span of time. The act of recognising changes in the environment around us is very crucial to us. The cars passing by us on the road continuously alter their location and occasionally their speed and direction. To prevent accidents, we must be able to detect and react to any changes in the environment. Humans (and most animals) are therefore quite excellent at detecting such changes in general. But our understanding of how changes are identified is still in a state of incompleteness.

According to the standard definition, "change detection" refers to the process through which an observer notices that a change has occurred. As well as detecting and localising changes, these skills presumably entail distinct processes. It has proven to be challenging for humans to have full knowledge of the change. This is partly because human vision is attention specific and may completely miss changes outside the attention region. Change detection intuitions are frequently very incorrect, as well. Most people assume that they can readily perceive any change in front of them, as long as the change is significant enough. However, research on "change blindness" has shown that we often fail to notice changes that have been occurring for a long time or that are expected. It is a phenomenon that runs against our intuitions about how change detection should operate, and it is one that is hard to explain. However, we have learnt a lot about what change detection is or how it functions to have such paradoxical outcomes.

This has motivated ML researchers to think about change detection using machines which do not have any biased expectations. Thus, detecting the changed regions of a location or scene across multiple images taken at different times has gained immense popularity. Applications of such ML models are found in diverse disciplines, including remote sensing, automatic video surveillance, medical image prognosis, civil engineering, and oceanography. "Change mask" refers to the collection of pixels that are "substantially different" from one picture to another in a timeline. When an item appears or disappears, or when it moves in relation to its surroundings, a change mask may be created by a combination of variables. Objects that are fixed can also vary in brightness or colour as a result of external influences.

Change detection in remotely sensed imagery is an area that has grown in prominence. Despite the fact that much work has been done in this area, it remains a busy and exciting field of research. The method of spotting distinct changes on the Earth's surface using remotely sensed images has been achieved through this method. The alterations can be both natural and induced by humans and hold crucial importance for various applications like military surveillance, agricultural study, environmental monitoring etc. Change detection algorithms focus on colour, texture, and/or shape features. These algorithms, however, are not precise enough to detect all sorts of changes in every image. This is a highly hard undertaking since there are several elements that must be considered in order for it to perform successfully. More integrated techniques like DNNs, intuitive pre-processing and post-processing, resilient statistics and meta-heuristic techniques are expected to drive future advancements. Segmentation, land-use monitoring, etc are only a few of the linked fields of research that will benefit from these future advances.

7.1.3 Chapter 5

Slow gradual changes as stated above are hard for humans to comprehend correctly all the time. However, when the changes are chaotic, it is almost impossible for humans to predict the behaviour of the next state. This is particularly witnessed while observing changes in the financial market. It is both difficult and risky. Moreover, these changes occur so frequently that they are often termed 'chaotic'. Traders purchase and sell stocks depending on their intuition. This needs a lot of expertise and knowledge. Over ninety-five per cent of traders lose money [226]. The concepts of the Open, High, Low and Close price (OHLC) data indicate the price trend of specific financial products over a specific period of time. As a result, investors continue to make purchases and sales according to correct forecasts of OHLC data, earning profits as a result. This is why it is vital to forecast the OHLC data, which was also our focus in this chapter.

To make extra gains, one should acquire specific financial products at prices close to the projected lows, and then sell them at prices close to the predictions of their highs. Data from the OHLC system may also be used to create candlestick charts, which illustrate the power of demand and supply in financial markets, as well as market conditions and public emotions.

However, there is a limit to such predictive capabilities. The stock prices are heavily chaotic and dependent on many external factors. Other variables such as demonetization or mergers and acquisitions might also impact the stock price. These intangible variables make it difficult to foresee the trends in advance. Moreover, the issues with customer trust always plague the adoption of ML models. First and foremost, the clients demand to understand how ML predictions differ from other quantitative conventional methods. In addition, consumers need to be made aware that they will not be able to follow or understand the logic behind the system's results, as with classic econometric models. When it comes to critical financial decisions, clients always want to understand what factors drive the model's predictions. This demand has given rise to research on explainable AI. DL-based models are still at their nascent stage in the direction of explainable AI. This is one future direction of research that holds immense scope. This would imply transparent solutions and increased acceptability of AI in the critical decision-making paradigms of the practical world. People are now ready to accept AI solutions for many tasks and are using them without even knowing. This is a hopeful outlook for AI researchers.

7.1.4 Chapter 6

As people rely on data-driven methods, they are becoming more concerned about sharing their data too. Thus, there is a dire need to preserve privacy. Also, the data generated by a single source may be small but if the data from all the sources is aggregated, the volume becomes unmanageable for a centralised computing system. This is called 'small big data'. Leveraging the power of so much data is a lucrative opportunity for ML researchers but is also challenging. The privacy issues and limitations of centralised computing have urged researchers to move to decentralised computing where the data is not exchanged across networks. Research on federated learning (FL) has allowed the ML community to harness the power of big data by treating each localised data segment independently and each local device as a centralised computing platform. It would then aggregate the learnt parameters from the local devices into a centralised server. Thus, data privacy is ensured as well as the model is learnt on the global data without the complexity of data transfer.

However, the heterogeneity of datasets across the local devices is often a challenge. The local datasets are not a proper representation of the global data. Thus, local models learnt on local datasets are often different from each other. Aggregation of such different models is often erroneous. Also, there are practical issues such as concept drift, communication expenses, scalability, etc which still have hindered FL to be accepted in practicality.

Future holds great hopes for DNNs trained on such a massive scale of datasets through

FL. There are some key considerations which need to be made before that. When it comes to multi-device federated learning, it must be ensured that devices have the capacity to train models. Also, the number of local models that can be parallelly learnt through FL still has a limit. In future, we may hope for better algorithms which may help increase the scalability.

7.2 Discussions

The thesis tried to handle a variety of pattern recognition challenges that come with the datasets. Within the framework of the research conducted, AEs (AEs) are particularly focused. However, as already mentioned in Chapter 1, AE is not a panacea and there are obviously many limitations which they suffer from. Vanilla AEs (AEs with a single hidden layer) might reduce the dimensions without the complexity of vanishing and exploding gradients, but they are not as efficient as the deep AEs. However, deep AEs on the other hand might again give rise to issues which arise with depth. There is still a lack of research on the limit to which this depth might be permissible. Residual networks [141] with skip connections are proposed as an alternative for deeper architectures, but the research is still in its nascent stages.

As for AEs, the bottleneck dimension is again another hyperparameter that greatly affects the performance. The important features/ information might get lost due to the lower dimension of the bottleneck layer. This problem might be avoided by testing reconstruction accuracy by varying dimensions of the bottleneck layer. A trade-off has to be done between the varying dimension of the bottleneck layer and reproduction loss.

AEs being trained in an unsupervised fashion require a huge amount of unlabelled data. Thus, to develop a robust AE, a large amount of training data is required. However, possible solutions like data augmentation can solve this issue by a substantial amount. Along with the huge amount of data, the data should be relevant to make the model generalize properly to the required application for which the model is designed.

Compression against conceptualization is a typical challenge for AEs. We want AEs to learn concepts rather than simply compressing data, yet the optimisation we do is to reduce the error of reconstruction. A proper theoretical understanding of AEs needs to be explored. Algorithms need to be devised which would help an AE differentiate between relevant and irrelevant features. This has given rise to some preliminary research regarding the limited use of labels in training an AE [227, 228]. In such cases, the issue of reducing the error of reconstruction may be augmented with the available label information. This would ensure that only relevant features are extracted.

AE-based data compression is learnt and is lossy. This makes their use limited. They cannot be applied to the use cases where compression degradation affects system performance in a significant way. Moreover, the decoder process is never perfect. A decision has to be taken about how much loss is tolerable for the use case. Since it is unsupervised, identification of relevant or important variables becomes critical. This is observed in the experiments done in Chapter 4 section 4.3.2. However, there is a desirable property of AE that the mappings produced by an AE are distinct. This is why AEs can be more sensitive to input errors than manual approaches, which might sometimes be desirable as in the case of outlier detection shown in Chapter 3.

AEs have another property which makes them desirable. It is the ability of transfer learning [229, 230]. Since the AE is learnt in an unsupervised manner, any pre-trained AE trained using data of a similar domain might be stacked on top of a classifier and be re-trained for a related domain. However, there is no guarantee that the features common between the two domains may be interpretable.

In this thesis, it is shown that AEs are well suited for certain tasks of pattern recognition. However, there are other tasks for which a different variety of AE may be much more suitable. There are several existing AE architectures which are not explored in this study.

7.3 Directions Unexplored

In the due course of the research, various interesting AE architectures have emerged which have their own strengths and shortcomings. The thesis has not explored the generative models of AE like Generative Adversarial Networks (GANs) [47] and Variational autoencoders (VAEs) [46]. GANs and VAEs fall under the category of a deep generative model which is used to generate realistic data like images, text, or sound artificially.

Generative modelling is an unsupervised learning job that incorporates the automated discovery and learning of the behaviour patterns or structures of input data utilised to create new instances in such a manner that might seem conceivably be taken from the original data.

GANs are one of the wisest approaches to training a generative model by approaching the task using two sub-models with a supervised training matter: the generator model designed in order to create fresh examples, and the discriminatory model which attempts to identify instances as real or fraudulent (generated). The two sub-models are trained in an adversarial zero-sum game until the discriminatory model is mistaken for about half the time, which means that the generator creates believable instances.

GANs provide the promise of generative models in order to produce realistic instances across a number of problem domains, especially in picture-to-picture translation tasks such as transforming pictures from summer to winter or day to night and producing photorealistic photos of items, settings and humans `https://thispersondoesnotexist.com/` that even humans cannot tell are machine-generated and not actually photographed.

VAEs belong to the families of probabilistic graphical models and variational Bayesian methods. Here, the encoder compresses the input data into a constrained multivariate latent distribution (encoder) to enhance the accuracy of the reconstruction (decoding).

A VAE is an AE where during training of the encoder, the distribution is regularized to ensure that the latent space has required features that are capable to generate new data. A VAE uses probability to describe an observation in latent space. So, instead of developing an encoder which produces a single value as output to describe each latent state attribute, the encoder is formulated to describe a probability distribution for each latent attribute. For VAEs, the encoder model is sometimes referred to as the recognition model whereas the decoder model is sometimes referred to as the generative model.

It differs from the normal AE where the input data gets converted into an encoded vector where each dimension represents some learned attributes representing the given data but in VAE, the encoder model's output is comprised of parameters that describes a distribution for each dimension of the latent space.

There are other AE models being developed like LSTM-AEs [231] which are currently used for sequential modelling. For sequential data, an LSTM-AE is created to read the input sequence, encode it, decode it, and recreate it. These types of AEs work well with datasets having temporal dependencies.

Each AE variant is built with a specific problem in mind. As the 'No Free Lunch Theorem' states, there is no algorithm which might be deemed as the best of all for all possible problems in the world. This is because every algorithm makes some assumptions regarding approaching a specific problem. These assumptions sometimes may suit the problem, in which case the algorithm outperforms others and sometimes may not suit the problem in which case the algorithm will fail.

7.4 Future Scope

The work of the thesis mainly revolves around the property of AE to map atypical patterns to different values. In that regard, we plan to look into the practical implementations of this property. Since this thesis has proposed an AE-supplemented outlier detection mechanism, it might be extended in future to modern IoT scenarios like smart homes, smart cities, etc. The data generated in such cases is extremely varied as these IoT devices might be highly diverse. In addition, multiple network protocols might generate data in varied formats. These impact the characteristics of supervised and unsupervised outlier detection algorithms. The multi-class and multi-type categorization of diverse forms of anomalies will be hard and would require a multi-modal approach.

Change detection would also find great applications in such smart cities. Automatic inspection of landscapes would be much easier. However, these would again require real-time models which would learn in an unsupervised manner. In future, we plan to adapt our proposed FFCAE model for such real-life problems.

While talking about real-life problems, only designing smart infrastructure won't suffice. One would need strong financial means to afford those technologies too. In that case, everyone would need smart financial planning. We intend to scale up the proposed financial forecasting model with real-time news and other market events too. The proposed model may also be adapted in future for crypto-currencies and forex trading. This would prove to be of substantial interest to investors and may pave the way for increased economic growth.

Lastly, if we wish to build these models for such a huge-scale practical scenario, we would need to think in the direction of distributed computing too. Blockchain concepts may be integrated into the proposed distributed deep learning paradigm in future. The research on AI is already marked by various disruptive technologies built on top of another. These developments will be further propelled by the reduced prices of processing units like GPGPUs (general purpose graphics processing units).

In general, the research on AEs is quite generic. AEs are not as commonly utilised in actual applications and if used, they only find applicability for denoising of data, reduction of dimensionality and as generative models for data augmentation. They are nonetheless extremely straightforward and may be utilised efficiently for the extraction of features. This is why they are most attractive for applications which do not have high-quality manual labelling available. There has been very limited research on the applicability of AEs for practical problems. They have always been used as a supplement to some other method. However, there needs to be a separate direction of research on the relevance of AEs in different domains of machine learning like self-supervised learning, active learning, reinforcement learning etc.

Publications

Journals

(a) D. Chakraborty, V. Narayanan, and A. Ghosh, "Integration of Deep Feature Extraction and Ensemble Learning for Outlier Detection," *Pattern Recognition*, vol. 89, pp. 161-171, 2019.

Conferences

(a) D. Chakraborty, D. Garg, A. Ghosh, and J. H. Chan, "Trigger Detection System for American Sign Language using Deep Convolutional Neural Networks," in *Proceedings of the 10th International Conference on Advances in Information Technology - IAIT 2018*, 2018.

Bibliography

[1] G. Cybenko, “Approximation by Superpositions of a Sigmoidal Function,” *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[2] K. Hornik, M. Stinchcombe, and H. White, “Multilayer Feedforward Networks are Universal Approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[3] J. Håstad, *Computational Limitations of Small-Depth Circuits.* Cambridge, MA, USA: MIT Press, 1987.

[4] J. Håstad and M. Goldmann, “On the Power of Small-Depth Threshold Circuits,” *Computational Complexity*, vol. 1, no. 2, pp. 113–129, 1991.

[5] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.* JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[6] Y. S. Abu-Mostafa, “The Vapnik-Chervonenkis Dimension: Information Versus Complexity in Learning,” *Neural Computation*, vol. 1, no. 3, pp. 312–317, 1989.

[7] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, “Why and When can Deep-but not Shallow-Networks avoid the Curse of Dimensionality: A Review,” *International Journal of Automation and Computing*, vol. 14, no. 5, pp. 503–519, 2017.

[8] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, “Extracting and Composing Robust Features with Denoising Autoencoders,” in *Proceedings of the 25th International Conference on Machine Learning.* ACM, 2008, pp. 1096–1103.

[9] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[10] G. E. Hinton, S. Osindero, and Y. W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[11] M. Ribeiro, A. E. Lazzaretti, and H. S. Lopes, "A Study of Deep Convolutional Auto-encoders for Anomaly Detection in Videos," *Pattern Recognition Letters*, vol. 105, pp. 13–22, 2018.

[12] L. Pasa and A. Sperduti, "Pre-training of Recurrent Neural Networks via Linear Autoencoders," *Advances in Neural Information Processing Systems*, vol. 27, pp. 3572–3580, 2014.

[13] M. F. Ferreira, R. Camacho, and L. F. Teixeira, "Using Autoencoders as a Weight Initialization Method on Deep Neural Networks for Disease Detection," *BMC Medical Informatics and Decision Making*, vol. 20, no. 5, pp. 1–18, 2020.

[14] E. Haber and L. Ruthotto, "Stable Architectures for Deep Neural Networks," *Inverse Problems*, vol. 34, no. 1, p. 014004, 2017.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Computation and Machine Learning Series. MIT Press, 2016. [Online]. Available: https://books.google.co.in/books?id=Np9SDQAAQBAJ

[16] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based Dimensionality Reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[17] D. Chakraborty, V. Narayanan, and A. Ghosh, "Integration of Deep Feature Extraction and Ensemble Learning for Outlier Detection," *Pattern Recognition*, vol. 89, pp. 161–171, 2019.

[18] J. E. Dayhoff and J. M. DeLeo, "Artificial Neural Networks: Opening the Black Box," *Cancer: Interdisciplinary International Journal of the American Cancer Society*, vol. 91, no. S8, pp. 1615–1635, 2001.

[19] H. C. Shin, M. R. Orton, D. J. Collins, S. J. Doran, and M. O. Leach, "Stacked Autoencoders for Unsupervised Feature Learning and Multiple Organ Detection in a Pilot Study using 4D Patient Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1930–1943, 2012.

[20] T. Amarbayasgalan, B. Jargalsaikhan, and K. H. Ryu, "Unsupervised Novelty Detection using Deep Autoencoders with Density based Clustering," *Applied Sciences*, vol. 8, no. 9, p. 1468, 2018.

[21] A. Majumdar, "Kernelized Linear Autoencoder," *Neural Processing Letters*, vol. 53, no. 2, pp. 1597–1614, 2021.

[22] Y. Chauvin and D. E. Rumelhart, *Backpropagation: Theory, Architectures, and Applications*, ser. Developments in Connectionist Theory Series. Taylor & Francis, 2013. [Online]. Available: https://books.google.co.in/books?id=B71nu3LDpREC

[23] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[24] G. Liu, Y. Li, L. Jiao, Y. Chen, and R. Shang, "Multiobjective Evolutionary Algorithm Assisted Stacked Autoencoder for PolSAR Image Classification," *Swarm and Evolutionary Computation*, vol. 60, p. 100794, 2021.

[25] Y. Lecun, "Modeles Connexionnistes de L'apprentissage (Connectionist Learning Models)(Ph. D. Thesis)," *IAAI Laboratory, Paris, France.*, 1987.

[26] G. W. Cottrell, P. Munro, and D. Zipser, "Learning Internal Representations from Gray-Scale Images: An Example of Extensional Programming," in *Proceedings Ninth Annual Conference of the Cognitive Science Society, Irvine, CA.*, 1985.

[27] P. Sibi, S. A. Jones, and P. Siddarth, "Analysis of Different Activation Functions using Back Propagation Neural Networks," *Journal of Theoretical and Applied Information Technology*, vol. 47, no. 3, pp. 1264–1268, 2013.

[28] P. Baldi and Z. Lu, "Complex-Valued Autoencoders," *Neural Networks*, vol. 33, pp. 136–147, 2012.

[29] M. Ringnér, "What is Principal Component Analysis?" *Nature Biotechnology*, vol. 26, no. 3, pp. 303–304, 2008.

[30] M. A. Kramer, "Nonlinear Principal Component Analysis using Autoassociative Neural Networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[31] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality Reduction: A Comparative Review," *Journal of Machine Learning Research*, vol. 10, no. 66-71, p. 13, 2009.

[32] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[33] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy Layer-Wise Training of Deep Networks," *Advances in Neural Information Processing Systems*, vol. 19, p. 153, 2007.

[34] E. Protopapadakis, A. Voulodimos, A. Doulamis, N. Doulamis, D. Dres, and M. Bimpas, "Stacked Autoencoders for Outlier Detection in Over-the-Horizon Radar Signals," *Computational Intelligence and Neuroscience*, vol. 2017, 2017.

[35] J. An and S. Cho, "Variational Autoencoder based Anomaly Detection using Reconstruction Probability," *SNU Data Mining Center, Technical Report*, 2015.

[36] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-Dimensional and Large-Scale Anomaly Detection using a Linear One-Class SVM with Deep Learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[37] B. Yan and G. Han, "Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System," *IEEE Access*, vol. 6, pp. 41 238–41 248, 2018.

[38] P. K. Mallick, S. H. Ryu, S. K. Satapathy, S. Mishra, G. N. Nguyen, and P. Tiwari, "Brain MRI Image Classification for Cancer Detection using Deep Wavelet Autoencoder-based Deep Neural Network," *IEEE Access*, vol. 7, pp. 46 278–46 287, 2019.

[39] M. Polic, I. Krajacic, N. Lepora, and M. Orsag, “Convolutional Autoencoder for Feature Extraction in Tactile Sensing,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3671–3678, 2019.

[40] A. Zhavoronkov, Y. A. Ivanenkov, A. Aliper, M. S. Veselov, V. A. Aladinskiy, A. V. Aladinskaya, V. A. Terentiev, D. A. Polykovskiy, M. D. Kuznetsov, A. Asadulaev *et al.*, “Deep Learning Enables Rapid Identification of Potent DDR1 Kinase Inhibitors,” *Nature Biotechnology*, vol. 37, no. 9, pp. 1038–1040, 2019.

[41] T. Poggio and L. Anselmi, F.and Rosasco, “I-theory on Depth vs Width: Hierarchical Function Composition,” Center for Brains, Minds and Machines (CBMM), Tech. Rep., 2015.

[42] K. G. Lore, A. Akintayo, and S. Sarkar, “LLNet: A Deep Autoencoder Approach to Natural Low-Light Image Enhancement,” *Pattern Recognition*, vol. 61, pp. 650–662, 2017.

[43] L. Meng, S. Ding, N. Zhang, and J. Zhang, “Research of Stacked Denoising Sparse Autoencoder,” *Neural Computing and Applications*, vol. 30, no. 7, pp. 2083–2100, 2018.

[44] T. Tanaka, “Mean-field Theory of Boltzmann Machine Learning,” *Physical Review E*, vol. 58, no. 2, p. 2302, 1998.

[45] R. Salakhutdinov, A. Mnih, and G. E. Hinton, “Restricted Boltzmann Machines for Collaborative Filtering,” in *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 791–798.

[46] S. J. Wetzel, “Unsupervised Learning of Phase Transitions: From Principal Component Analysis to Variational Autoencoders,” *Physical Review E*, vol. 96, no. 2, p. 022140, 2017.

[47] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, “Generative Adversarial Networks: An Overview,” *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[48] A. I. Zayed, "A Convolution and Product Theorem for the Fractional Fourier Transform," *IEEE Signal Processing Letters*, vol. 5, no. 4, pp. 101–103, 1998.

[49] D. Chakraborty, D. Garg, A. Ghosh, and J. H. Chan, "Trigger Detection System for American Sign Language using Deep Convolutional Neural Networks," in *Proceedings of the 10th International Conference on Advances in Information Technology*, 2018, pp. 1–6.

[50] K. Sohn, D. Y. Jung, H. Lee, and A. O. Hero, "Efficient Learning of Sparse, Distributed, Convolutional Feature Representations for Object Recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on.* IEEE, 2011, pp. 2643–2650.

[51] Y. Zhan, K. Fu, M. Yan, X. Sun, H. Wang, and X. Qiu, "Change Detection based on Deep Siamese Convolutional Network for Optical Aerial Images," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 10, pp. 1845–1849, 2017.

[52] Y. Xiong and R. Zuo, "Robust Feature Extraction for Geochemical Anomaly Recognition using a Stacked Convolutional Denoising Autoencoder," *Mathematical Geosciences*, pp. 1–22, 2021.

[53] J. K. Chow, Z. Su, J. Wu, P. S. Tan, X. Mao, and Y. H. Wang, "Anomaly Detection of Defects on Concrete Structures with the Convolutional Autoencoder," *Advanced Engineering Informatics*, vol. 45, p. 101105, 2020.

[54] Y. Lin and J. Wang, "Probabilistic Deep Autoencoder for Power System Measurement Outlier Detection and Reconstruction," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1796–1798, 2019.

[55] M. J. Kurian and R. S. Gladston, "Improving the Performance of a Classification based Outlier Detection System using Dimensionality Reduction Techniques," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 9, 2017.

[56] C. Zhao, H. Cheng, and S. Feng, "A Spectral-Spatial Change Detection Method Based on Simplified 3-D Convolutional Autoencoder for Multitemporal Hyperspectral Images," *IEEE Geoscience and Remote Sensing Letters*, 2021.

[57] S. Liu, L. Bruzzone, F. Bovolo, and P. Du, "Hierarchical Unsupervised Change Detection in Multitemporal Hyperspectral Images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 1, pp. 244–260, 2014.

[58] Y. K. Han, A. Chang, S. K. Choi, H. Park, and J. Choi, "An Unsupervised Algorithm for Change Detection in Hyperspectral Remote Sensing Data using Synthetically Fused Images and Derivative Spectral Profiles," *Journal of Sensors*, vol. 2017, 2017.

[59] Q. Wang, Z. Yuan, Q. Du, and X. Li, "GETNET: A General End-to-end 2-d CNN Framework for Hyperspectral Image Change Detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 1, pp. 3–13, 2018.

[60] H. Wu and S. Prasad, "Semi-supervised Deep Learning using Pseudo Labels for Hyperspectral Image Classification," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1259–1270, 2017.

[61] F. De Morsier, D. Tuia, M. Borgeaud, V. Gass, and J. P. Thiran, "Semi-supervised Novelty Detection using SVM Entire Solution Path," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 4, pp. 1939–1950, 2013.

[62] M. Roy, S. Ghosh, and A. Ghosh, "A Neural Approach under Active Learning Mode for Change Detection in Remotely Sensed Images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 4, pp. 1200–1206, 2013.

[63] J. Zahedi and M. M. Rounaghi, "Application of Artificial Neural Network Models and Principal Component Analysis Method in Predicting Stock Prices on Tehran Stock Exchange," *Physica A: Statistical Mechanics and its Applications*, vol. 438, pp. 178–187, 2015.

[64] D. Erhan, Y. Bengio, A. Courville, P. Manzagol, P. Vincent, and S. Bengio, "Why Does Unsupervised Pre-Training Help Deep Learning?" *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.

[65] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 52–59.

[66] T. Hastie and W. Stuetzle, "Principal Curves," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 502–516, 1989.

[67] B. Schölkopf, A. Smola, and K. R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[68] L. Van der Maaten and G. Hinton, "Visualizing Data using t-SNE." *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.

[69] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[70] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[71] S. S. Vempala, *The Random Projection Method*. American Mathematical Soc., 2005, vol. 65.

[72] K. Vu, P. L. Poirion, and L. Liberti, "Gaussian Random Projections for Euclidean Membership Problems," *Discrete Applied Mathematics*, vol. 253, pp. 93–102, 2019.

[73] D. Hawkins, *Identification of Outliers*. Springer Netherlands, 2014. [Online]. Available: https://books.google.co.in/books?id=toR6oAEACAAJ

[74] A. Ghosh, "Big Data and its Utility," *Consulting Ahead*, vol. 10, no. 1, pp. 52–68, 2016.

[75] A. Amin, S. Anwar, A. Adnan, M. Nawaz, N. Howard, J. Qadir, A. Hawalah, and A. Hussain, "Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study," *IEEE Access*, vol. 4, pp. 7940–7957, 2016.

[76] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, “A Comparative Evaluation of Outlier Detection Algorithms: Experiments and Analyses,” *Pattern Recognition*, vol. 74, pp. 406–421, 2018.

[77] G. Attarde and A. Deshpande, “Outlier Detection Using Unsupervised and Semi-Supervised Technique on High Dimensional Data,” *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 5, no. 7, pp. 14 180–14 185, 2016.

[78] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, “On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study,” *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891–927, 2016.

[79] M. Radovanović, A. Nanopoulos, and M. Ivanović, “Reverse Nearest Neighbors in Unsupervised Distance-Based Outlier Detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1369–1382, 2015.

[80] K. Zhang, M. Hutter, and H. Jin, “A New Local Distance-Based Outlier Detection Approach for Scattered Real-World Data,” *Advances in Knowledge Discovery and Data Mining*, pp. 813–822, 2009.

[81] L. Zhang, J. Lin, and R. Karim, “Adaptive Kernel Density-Based Anomaly Detection for Nonlinear Systems,” *Knowledge-Based Systems*, vol. 139, pp. 50–63, 2018.

[82] C. Désir, S. Bernard, C. Petitjean, and L. Heutte, “One Class Random Forests,” *Pattern Recognition*, vol. 46, no. 12, pp. 3490–3506, 2013.

[83] D. M. J. Tax and R. P. W. Duin, “Support Vector Data Description,” *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.

[84] Q. Liu, R. Klucik, C. Chen, G. Grant, D. Gallaher, Q. Lv, and L. Shang, “Unsupervised Detection of Contextual Anomaly in Remotely Sensed Data,” *Remote Sensing of Environment*, vol. 202, pp. 75–87, 2017.

[85] X. Wang, X. L. Wang, Y. Ma, and D. M. Wilkes, “A fast MST-Inspired kNN-Based Outlier Detection Method,” *Information Systems*, vol. 48, pp. 89–112, 2015.

[86] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[87] Y. Chen, D. Miao, and H. Zhang, "Neighborhood Outlier Detection," *Expert Systems with Applications*, vol. 37, no. 12, pp. 8745–8749, 2010.

[88] Z. G. Liu, Q. Pan, and J. Dezert, "A New Belief-Based k-Nearest Neighbor Classification Method," *Pattern Recognition*, vol. 46, no. 3, pp. 834–844, 2013.

[89] G. Bhattacharya, K. Ghosh, and A. S. Chowdhury, "kNN Classification with an Outlier Informative Distance Measure," in *International Conference on Pattern Recognition and Machine Intelligence.* Springer, 2017, pp. 21–27.

[90] M. S. Sadooghi and S. E. Khadem, "Improving One Class Support Vector Machine Novelty Detection Scheme using Nonlinear Features," *Pattern Recognition*, 2018.

[91] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation-Based Anomaly Detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, p. 3, 2012.

[92] A. Ghosh, N. R. Pal, and S. K. Pal, "Self-Organization for Object Extraction using a Multilayer Neural Network and Fuzziness Measures," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 1, p. 54, 1993.

[93] A. L. Oliveira, F. R. Costa, and C. O. Filho, "Novelty Detection with Constructive Probabilistic Neural Networks," *Neurocomputing*, vol. 71, no. 4-6, pp. 1046–1053, 2008.

[94] V. Jumutc and J. A. K. Suykens, "Multi-Class Supervised Novelty Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2510–2523, 2014.

[95] A. Mellor, S. Boukir, A. Haywood, and S. Jones, "Exploring Issues of Training Data Imbalance and Mislabelling on Random Forest Performance for Large Area Land Cover Classification using the Ensemble Margin," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 155–168, 2015.

[96] A. Romero, C. Gatta, and G. Camps-Valls, “Unsupervised Deep Feature Extraction for Remote Sensing Image Classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1349–1362, 2016.

[97] C. C. Olson, K. P. Judd, and J. M. Nichols, “Manifold Learning Techniques for Unsupervised Anomaly Detection,” *Expert Systems with Applications*, vol. 91, pp. 374–385, 2018.

[98] S. Sathe and C. C. Aggarwal, “LODES: Local Density Meets Spectral Outlier Detection,” in *Proceedings of the 2016 SIAM International Conference on Data Mining.* SIAM, 2016, pp. 171–179.

[99] C. C. Aggarwal and S. Sathe, “Theoretical Foundations and Algorithms for Outlier Ensembles,” *ACM SIGKDD Explorations Newsletter*, vol. 17, no. 1, pp. 24–47, 2015.

[100] F. T. Liu, K. M. Ting, and Z. H. Zhou, “Isolation forest,” in *Eighth IEEE International Conference on Data Mining (ICDM), 2008.* IEEE, 2008, pp. 413–422.

[101] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, and J. R. Wells, “Efficient Anomaly Detection by Isolation using Nearest Neighbour Ensemble,” in *IEEE International Conference on Data Mining Workshop (ICDMW), 2014.* IEEE, 2014, pp. 698–705.

[102] R. Lyon, B. Stappers, S. Cooper, J. Brooke, and J. Knowles, “Fifty Years of Pulsar Candidate Selection: From Simple Filters to a New Principled Real-Time Classification Approach,” *Monthly Notices of the Royal Astronomical Society*, vol. 459, no. 1, pp. 1104–1123, 2016.

[103] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, “Calibrating Probability with Undersampling for Unbalanced Classification,” in *IEEE Symposium Series on Computational Intelligence, 2015.* IEEE, 2015, pp. 159–166.

[104] B. Micenková, B. McWilliams, and I. Assent, “Learning Outlier Ensembles: The Best of Both Worlds - Supervised and Unsupervised,” in *Proceedings of the ACM SIGKDD 2014 Workshop on Outlier Detection and Description under Data Diversity (ODD2). New York, NY, USA*, 2014, pp. 51–54.

[105] P. R. Nicolas, *Scala for Machine Learning: Data processing, ML algorithms, Smart Analytics, and more.* Packt Publishing, 2017. [Online]. Available: https://books.google.co.in/books?id=9plGDwAAQBAJ

[106] J. M. Bland and D. G. Altman, "Multiple Significance Tests: The Bonferroni Method," *The BMJ*, vol. 310, no. 6973, p. 170, 1995.

[107] D. Yu and L. Deng, *Automatic Speech Recognition: A Deep Learning Approach*, ser. Signals and Communication Technology. Springer London, 2014. [Online]. Available: https://books.google.co.in/books?id=rUBTBQAAQBAJ

[108] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2015.

[109] E. Ohn-Bar and M. M. Trivedi, "Hand Gesture Recognition in Real Time for Automotive Interfaces: A Multimodal Vision-Based Approach and Evaluations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2368–2377, 2014.

[110] H. L. Lane and F. Grosjean, *Recent Perspectives on American Sign Language.* Psychology Press, 2017. [Online]. Available: https://books.google.co.in/books?id=Fcg3DwAAQBAJ

[111] Kaggle. (2017) Sign Language MNIST: Drop-In Replacement for MNIST for Hand Gesture Recognition Tasks. [Online]. Available: https://www.kaggle.com/datamunge/sign-language-mnist

[112] A. M. Lindahl, "Speech Recognition Wake-Up of a Handheld Portable Electronic Device," Jan. 26 2016, uS Patent 9,245,527.

[113] S. W. Salvador, J. P. Lilly, F. V. Weber, J. P. Adams, and R. P. Thomas, "Wake Word Evaluation," Mar. 1 2016, uS Patent 9,275,637.

[114] G. Naithani, J. Kivinummi, T. Virtanen, O. Tammela, M. J. Peltola, and J. M. Leppänen, "Automatic Segmentation of Infant Cry Signals using Hidden Markov

Models," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2018, no. 1, p. 1, 2018.

[115] F. Ge and Y. Yan, "Deep Neural Network based Wake-Up-Word Speech Recognition with Two-Stage Detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.* IEEE, 2017, pp. 2761–2765.

[116] S. H. K. Parthasarathi, B. Hoffmeister, B. King, and R. Maas, "Anchored Speech Detection and Speech Recognition," Sep. 21 2017, uS Patent App. 15/196,228.

[117] J. L. Raheja, A. Mishra, and A. Chaudhary, "Indian Sign Language Recognition using SVM," *Pattern Recognition and Image Analysis*, vol. 26, no. 2, pp. 434–441, 2016.

[118] S. Nowozin, P. Kohli, and J. D. J. Shotton, "Gesture Detection and Recognition," Apr. 11 2017, uS Patent 9,619,035.

[119] M. Lech, B. Kostek, and A. Czyżewski, "Examining Classifiers Applied to Static Hand Gesture Recognition in Novel Sound Mixing System," in *Multimedia and Internet Systems: Theory and Practice*, ser. Advances in Intelligent Systems and Computing. Springer, 2013, pp. 77–86.

[120] L. Pigou, S. Dieleman, P. J. Kindermans, and B. Schrauwen, "Sign Language Recognition using Convolutional Neural Networks," in *Workshop at the European Conference on Computer Vision.* Springer, 2014, pp. 572–578.

[121] J. Huang, W. Zhou, H. Li, and W. Li, "Sign Language Recognition using 3D Convolutional Neural Networks," in *IEEE International Conference on Multimedia and Expo (ICME), 2015.* IEEE, 2015, pp. 1–6.

[122] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand Gesture Recognition with 3D Convolutional Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 1–7.

[123] X. Jin, Y. Gu, and T. Liu, "Intrinsic Image Recovery from Remote Sensing Hyperspectral Images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 1, pp. 224–238, 2018.

[124] M. Volpi, D. Tuia, F. Bovolo, M. Kanevski, and L. Bruzzone, “Supervised Change Detection in VHR Images using Contextual Information and Support Vector Machines,” *International Journal of Applied Earth Observation and Geoinformation*, vol. 20, pp. 77–85, 2013.

[125] Y. Yuan, H. Lv, and X. Lu, “Semi-supervised Change Detection Method for Multi-temporal Hyperspectral Images,” *Neurocomputing*, vol. 148, pp. 363–375, 2015.

[126] B. Du, L. Ru, C. Wu, and L. Zhang, “Unsupervised Deep Slow Feature Analysis for Change Detection in Multi-temporal Remote Sensing Images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 12, pp. 9976–9992, 2019.

[127] S. Ghosh, L. Bruzzone, S. Patra, F. Bovolo, and A. Ghosh, “A Context-Sensitive Technique for Unsupervised Change Detection based on Hopfield-type Neural Networks,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 3, pp. 778–789, 2007.

[128] M. Zanetti and L. Bruzzone, “A Theoretical Framework for Change Detection based on a Compound Multiclass Statistical Model of the Difference Image,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 1129–1143, 2017.

[129] R. S. Dewi, W. Bijker, and A. Stein, “Change Vector Analysis to Monitor the Changes in Fuzzy Shorelines,” *Remote Sensing*, vol. 9, no. 2, p. 147, 2017.

[130] S. Rahman and V. Mesev, “Change Vector Analysis, Tasseled Cap, and NDVI-NDMI for Measuring Land Use/ Cover Changes Caused by a Sudden Short-Term Severe Drought: 2011 Texas Event,” *Remote Sensing*, vol. 11, no. 19, p. 2217, 2019.

[131] G. Gao, M. Wang, X.and Niu, and S. Zhou, “Modified Log-ratio Operator for Change Detection of Synthetic Aperture Radar Targets in Forest Concealment,” *Journal of Applied Remote Sensing*, vol. 8, no. 1, p. 083583, 2014.

[132] T. Celik, “Unsupervised Change Detection in Satellite Images using Principal Component Analysis and k-means Clustering,” *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 4, pp. 772–776, 2009.

[133] A. A. Nielsen, “The Regularized Iteratively Reweighted MAD Method for Change Detection in Multi-and Hyperspectral Data,” *IEEE Transactions on Image processing*, vol. 16, no. 2, pp. 463–478, 2007.

[134] B. Wang, S. K. Choi, Y. K. Han, S. K. Lee, and J. W. Choi, “Application of IR-MAD using Synthetically Fused Images for Change Detection in Hyperspectral Data,” *Remote Sensing Letters*, vol. 6, no. 8, pp. 578–586, 2015.

[135] O. A. Carvalho Júnior, R. F. Guimarães, A. R. Gillespie, N. C. Silva, and R. A. T. Gomes, “A New Approach to Change Vector Analysis using Distance and Similarity Measures,” *Remote Sensing*, vol. 3, no. 11, pp. 2473–2493, 2011.

[136] A. M. G. Tommaselli, L. D. Santos, R. A. de Oliveira, A. Berveglieri, N. N. Imai, and E. Honkavaara, “Refining the Interior Orientation of a Hyperspectral Frame Camera with Preliminary Bands Co-Registration,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 7, pp. 2097–2106, 2019.

[137] M. Hasanlou and S. T. Seydi, “Hyperspectral Change Detection: An Experimental Comparative Study,” *International Journal of Remote Sensing*, vol. 39, no. 20, pp. 7029–7083, 2018.

[138] J. L. Fandino, A. S. Garea, D. B. Heras, and F. Argüello, “Stacked Autoencoders for Multiclass Change Detection in Hyperspectral Images,” in *International Geoscience and Remote Sensing Symposium IGARSS 2018*. IEEE, 2018, pp. 1906–1909. [Online]. Available: http://dx.doi.org/10.1109/IGARSS.2018.8518338

[139] H. Abdi and L. J. Williams, “Tukey’s Honestly Significant Difference (HSD) Test,” *Encyclopedia of Research Design*, vol. 3, no. 1, pp. 1–5, 2010.

[140] G. Schohn and D. Cohn, “Less is More: Active Learning with Support Vector Machines,” in *International Conference on Machine Learning*, vol. 2, no. 4. Citeseer, 2000, p. 6.

[141] Y. Li, C. Peng, Y. Chen, L. Jiao, L. Zhou, and R. Shang, “A Deep Learning Method for Change Detection in Synthetic Aperture Radar Images,” *IEEE Transactions on*

Geoscience and Remote Sensing, vol. 57, no. 8, pp. 5751–5763, 2019.

[142] A. Pozdnoukhov and M. Kanevski, “Monitoring Network Optimisation for Spatial Data Classification using Support Vector Machines,” *International Journal of Environment and Pollution*, vol. 28, no. 3-4, pp. 465–484, 2006.

[143] C. Geiß, M. Thoma, and H. Taubenböck, “Cost-Sensitive Multitask Active Learning for Characterization of Urban Environments with Remote Sensing,” *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 6, pp. 922–926, 2018.

[144] T. Luo, K. Kramer, D. B. Goldgof, L. O. Hall, S. Samson, A. Remsen, T. Hopkins, and D. Cohn, “Active Learning to Recognize Multiple Types of Plankton.” *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[145] J. Wu, V. S. Sheng, J. Zhang, H. Li, T. Dadakova, C. L. Swisher, Z. Cui, and P. Zhao, “Multi-Label Active Learning Algorithms for Image Classification: Overview and Future Promise,” *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–35, 2020.

[146] J. M. Haut, M. E. Paoletti, J. Plaza, J. Li, and A. Plaza, “Active Learning with Convolutional Neural Networks for Hyperspectral Image Classification using a New Bayesian Approach,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 11, pp. 6440–6461, 2018.

[147] S. Piramanayagam, E. Saber, W. Schwartzkopf, and F. W. Koehler, “Supervised Classification of Multisensor Remotely Sensed Images using a Deep Learning Framework,” *Remote Sensing*, vol. 10, no. 9, p. 1429, 2018.

[148] V. Ruzicka, S. D’Aronco, J. D. Wegner, and K. Schindler, “Deep Active Learning in Remote Sensing for Data Efficient Change Detection,” in *Proceedings of MACLEAN: MAChine Learning for EArth ObservatioN Workshop co-located with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2020)*, vol. 2766. RWTH Aachen University, 2020.

[149] J. L. Fandino, D. B. Heras, F. Argüello, and M. D. Mura, "GPU Framework for Change Detection in Multitemporal Hyperspectral Images," in *10th International Symposium on High-Level Parallel Programming and Applications*, 2017, pp. 115 – 132.

[150] K. Adam, A. Marcet, and J. P. Nicolini, "Stock Market Volatility and Learning," *The Journal of Finance*, vol. 71, no. 1, pp. 33–82, 2016.

[151] K. De Bot, "Complexity Theory and Dynamic Systems Theory," *Complexity Theory and Language Development: In Celebration of Diane Larsen-Freeman*, vol. 48, p. 51, 2017.

[152] Y. J. Chen, Y. M. Chen, and C. L. Lu, "Enhancement of Stock Market Forecasting using an Improved Fundamental Analysis-based Approach," *Soft Computing*, vol. 21, no. 13, pp. 3735–3757, 2017.

[153] P. Tiwari, "Comparison of Different Natural Language Processing Models and Neural Networks for Predicting Stock Prices: A Case Study," *IEEE Transactions on Microwave Theory and Techniques*, vol. 60, no. 12, p. 1, 2012.

[154] R. D. Edwards, J. Magee, and W. H. C. Bassetti, *Technical Analysis of Stock Trends.* Taylor & Francis, 2018. [Online]. Available: https://books.google.co.in/books?id=cVJmDwAAQBAJ

[155] B. G. Malkiel, "The Efficient Market Hypothesis and its Critics," *Journal of Economic Perspectives*, vol. 17, no. 1, pp. 59–82, 2003.

[156] Y. S. Abu-Mostafa and A. F. Atiya, "Introduction to Financial Forecasting," *Applied Intelligence*, vol. 6, no. 3, pp. 205–213, 1996.

[157] Y. C. Tsai, M. E. Wu, J. H. Syu, C. L. Lei, C. S. Wu, J. M. Ho, and C. J. Wang, "Assessing the Profitability of Timely Opening Range Breakout on Index Futures Markets," *IEEE Access*, vol. 7, pp. 32 061–32 071, 2019.

[158] V. Karalevicius, N. Degrande, and J. De Weerdt, "Using Sentiment Analysis to Predict Interday Bitcoin Price Movements," *The Journal of Risk Finance*, vol. 19, no. 1, pp. 56–75, 2018.

[159] M. Ballings, D. Van den Poel, N. Hespeels, and R. Gryp, "Evaluating Multiple Classifiers for Stock Price Direction Prediction," *Expert Systems with Applications*, vol. 42, no. 20, pp. 7046–7056, 2015.

[160] R. Singh and S. Srivastava, "Stock Prediction using Deep Learning," *Multimedia Tools and Applications*, vol. 76, no. 18, pp. 18 569–18 584, 2017.

[161] H. Y. Kim and C. H. Won, "Forecasting the Volatility of Stock Price Index: A Hybrid Model Integrating LSTM with Multiple GARCH-type Models," *Expert Systems with Applications*, vol. 103, pp. 25–37, 2018.

[162] R. W. Colby, *The Encyclopedia of Technical Market Indicators, Second Edition*. McGraw-Hill Education, 2003. [Online]. Available: https://books.google.co.in/books?id=f82LUFQVGOUC

[163] W. Budiharto, "Data Science Approach to Stock Prices Forecasting in Indonesia during Covid-19 using Long Short-Term Memory (LSTM)," *Journal of big data*, vol. 8, no. 1, pp. 1–9, 2021.

[164] M. Gorenc Novak and D. Velušček, "Prediction of Stock Price Movement based on Daily High Prices," *Quantitative Finance*, vol. 16, no. 5, pp. 793–826, 2016.

[165] A. H. Manurung, W. Budiharto, and H. Prabowo, "Algorithm and Modeling of Stock Prices Forecasting based on Long Short-Term Memory (LSTM)," *International Journal of Innovative Computing Information and Control (ICIC)*, vol. 12, no. 12, pp. 1277–1283, 2018.

[166] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[167] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita, "Adaptive Stock Trading Strategies with Deep Reinforcement Learning Methods," *Information Sciences*, vol. 538, pp. 142–158, 2020.

[168] S. Panigrahi and H. S. Behera, "Effect of Normalization Techniques on Univariate Time Series Forecasting using Evolutionary Higher Order Neural Network,"

International Journal of Engineering and Advanced Technology, vol. 3, no. 2, pp. 280–285, 2013.

[169] T. Hussain, K. Muhammad, A. Ullah, Z. Cao, S. W. Baik, and V. H. C. de Albuquerque, "Cloud-assisted multiview video summarization using cnn and bidirectional lstm," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 77–86, 2019.

[170] E. F. Fama, "The Behavior of Stock-Market Prices," *The Journal of Business*, vol. 38, no. 1, pp. 34–105, 1965.

[171] R. K. Nayak, D. Mishra, and A. K. Rath, "A Naïve SVM-*k*NN based Stock Market Trend Reversal Analysis for Indian Benchmark Indices," *Applied Soft Computing*, vol. 35, pp. 670–680, 2015.

[172] P. Mondal, L. Shit, and S. Goswami, "Study of Effectiveness of Time Series modeling (ARIMA) in Forecasting Stock Prices," *International Journal of Computer Science, Engineering and Applications*, vol. 4, no. 2, p. 13, 2014.

[173] A. A. Adebiyi, A. O. Adewumi, and C. K. Ayo, "Comparison of ARIMA and Artificial Neural Networks Models for Stock Price Prediction," *Journal of Applied Mathematics*, vol. 2014, 2014.

[174] D. Enke and N. Mehdiyev, "Stock Market Prediction using a Combination of Stepwise Regression Analysis, Differential Evolution-based Fuzzy Clustering, and a Fuzzy Inference Neural Network," *Intelligent Automation & Soft Computing*, vol. 19, no. 4, pp. 636–648, 2013.

[175] X. Zhong and D. Enke, "Forecasting Daily Stock Market Return using Dimensionality Reduction," *Expert Systems with Applications*, vol. 67, pp. 126–139, 2017.

[176] F. M. Talarposhti, H. J. Sadaei, R. Enayatifar, F. G. Guimarães, M. Mahmud, and T. Eslami, "Stock Market Forecasting by using a Hybrid Model of Exponential Fuzzy Time Series," *International Journal of Approximate Reasoning*, vol. 70, pp. 79–98, 2016.

[177] M. Y. Chen and B. T. Chen, “A Hybrid Fuzzy Time Series Model based on Granular Computing for Stock Price Forecasting,” *Information Sciences*, vol. 294, pp. 227–241, 2015.

[178] B. M. Henrique, V. A. Sobreiro, and H. Kimura, “Stock Price Prediction using Support Vector Regression on Daily and Up to the Minute Prices,” *The Journal of Finance and Data Science*, vol. 4, no. 3, pp. 183–201, 2018.

[179] Y. Wang, H. Liu, Q. Guo, S. Xie, and X. Zhang, “Stock Volatility Prediction by Hybrid Neural Network,” *IEEE Access*, vol. 7, pp. 154 524–154 534, 2019.

[180] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[181] T. Fischer and C. Krauss, “Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions,” *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[182] C. Francq and J. M. Zakoian, *GARCH Models: Structure, Statistical Inference and Financial Applications.* John Wiley & Sons, 2019.

[183] G. Shen, Q. Tan, H. Zhang, P. Zeng, and J. Xu, “Deep Learning with Gated Recurrent Unit Networks for Financial Sequence Predictions,” *Procedia computer science*, vol. 131, pp. 895–903, 2018.

[184] S. Ahmed, S. U. Hassan, N. R. Aljohani, and R. Nawaz, “FLF-LSTM: A Novel Prediction System using Forex Loss Function,” *Applied Soft Computing*, vol. 97, p. 106780, 2020.

[185] S. Mootha, S. Sridhar, R. Seetharaman, and S. Chitrakala, “Stock Price Prediction using Bi-Directional LSTM based Sequence to Sequence Modeling and Multitask Learning,” in *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON).* IEEE, 2020, pp. 0078–0086.

[186] J. Baxter, “A Bayesian/ Information Theoretic Model of Learning to Learn via Multiple Task Sampling,” *Machine learning*, vol. 28, no. 1, pp. 7–39, 1997.

[187] O. Reyes and S. Ventura, “Performing Multi-Target Regression via a Parameter Sharing-based Deep Network,” *International journal of neural systems*, vol. 29, no. 09, p. 1950014, 2019.

[188] R. F. Engle, E. Ghysels, and B. Sohn, “Stock Market Volatility and Macroeconomic Fundamentals,” *Review of Economics and Statistics*, vol. 95, no. 3, pp. 776–797, 2013.

[189] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, “LSTM Network: A Deep Learning Approach for Short-Term Traffic Forecast,” *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.

[190] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[191] Y. Zhang and Q. Yang, “An Overview of Multi-Task Learning,” *National Science Review*, vol. 5, no. 1, pp. 30–43, 2018.

[192] J. Sola and J. Sevilla, “Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems,” *IEEE Transactions on Nuclear Science*, vol. 44, no. 3, pp. 1464–1468, 1997.

[193] S. Bhanja and A. Das, “Impact of Data Normalization on Deep Neural Network for Time Series Forecasting,” in *Proceedings of Conference on Advancement in Computation, Communication and Electronics Paradigm (ACCEP-2019)*, 2019, p. 27.

[194] K. Shibata and Y. Ikeda, “Effect of Number of Hidden Neurons on Learning in Large-Scale Layered Neural Networks,” in *2009 ICCAS-SICE*. IEEE, 2009, pp. 5008–5013.

[195] J. Xie, K. Hu, M. Zhu, and Y. Guo, “Bioacoustic Signal Classification in Continuous Recordings: Syllable-Segmentation vs Sliding-Window,” *Expert Systems with Applications*, vol. 152, p. 113390, 2020.

[196] R. Dash and P. K. Dash, “A Hybrid Stock Trading Framework Integrating Technical Analysis with Machine Learning Techniques,” *The Journal of Finance and Data Science*, vol. 2, no. 1, pp. 42–57, 2016.

[197] S. A. Wadi, M. Almasarweh, A. A. Alsaraireh, and J. Aqaba, "Predicting Closed Price Time Series Data using ARIMA Model," *Modern Applied Science*, vol. 12, no. 11, 2018.

[198] A. Suharsono, A. Aziza, and W. Pramesti, "Comparison of Vector Autoregressive (VAR) and Vector Error Correction Models (VECM) for Index of ASEAN Stock Price," in *AIP Conference Proceedings*, vol. 1913, no. 1. AIP Publishing LLC, 2017, p. 020032.

[199] O. Gupta and R. Raskar, "Distributed Learning of Deep Neural Network over Multiple Agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[200] M. Re and G. Valentini, "Ensemble Methods," *Advances in Machine Learning and Data Mining for Astronomy*, pp. 563–593, 2012.

[201] C. Zhang and Y. Ma, *Ensemble Machine Learning: Methods and Applications*. Springer, 2012.

[202] Z. Tamen, H. Drias, and D. Boughaci, "An Efficient Multiple Classifier System for Arabic Handwritten Words Recognition," *Pattern Recognition Letters*, vol. 93, pp. 123–132, 2017.

[203] X. Wu, X. Zhu, G. Q. Wu, and W. Ding, "Data Mining with Big Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, 2013.

[204] D. Sarkar, A. Narang, and S. Rai, "Fed-Focal Loss for Imbalanced Data Classification in Federated Learning," *arXiv preprint arXiv:2011.06283*, 2020.

[205] L. Wang, S. Xu, X. Wang, and Q. Zhu, "Addressing Class Imbalance in Federated Learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 11, 2021, pp. 10 165–10 173.

[206] Y. Zhou, Q. Ye, and J. C. Lv, "Communication-efficient federated learning with compensated overlap-fedavg," *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[207] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-Balancing Federated Learning with Global Imbalanced Data in Mobile Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59–71, 2020.

[208] K. Jacobs, "Independent Identically Distributed (iid) Random Variables," in *Discrete Stochastics.* Springer, 1992, pp. 65–101.

[209] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Artificial Intelligence and Statistics.* PMLR, 2017, pp. 1273–1282.

[210] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Transactions on Cognitive Communications and Networking*, 2021.

[211] Y. Liu, X. Li, X. Chen, X. Wang, and H. Li, "High-performance machine learning for large-scale data classification considering class imbalance," *Scientific Programming*, vol. 2020, 2020.

[212] A. Rosenberg and J. Hirschberg, "V-Measure: A Conditional Entropy-based External Cluster Evaluation Measure," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007, pp. 410–420.

[213] C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl, "Measuring the Effects of Data Parallelism on Neural Network Training," *Journal of Machine Learning Research*, vol. 20, no. 112, pp. 1–49, 2019.

[214] S. Moreno-Alvarez, J. M. Haut, M. E. Paoletti, and J. A. Rico-Gallego, "Heterogeneous Model Parallelism for Deep Neural Networks," *Neurocomputing*, vol. 441, pp. 1–12, 2021.

[215] Y. Oyama, N. Maruyama, N. Dryden, E. McCarthy, P. Harrington, J. Balewski, S. Matsuoka, P. Nugent, and B. Van Essen, "The Case for Strong Scaling in Deep Learning: Training Large 3D CNNs with Hybrid Parallelism," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1641–1652, 2020.

[216] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. Deng, "Privacy-Preserving Federated Deep Learning with Irregular Users," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[217] S. Zhang, A. Choromanska, and Y. Lecun, "Deep Learning with Elastic Averaging SGD," *Advances in Neural Information Processing Systems*, vol. 2015, pp. 685–693, 2015.

[218] L. Hyunsoo, S. Hong, J. Bang, and H. Kim, "A Study on Optimization Techniques for Applying Federated Learning to Class Imbalance Problems," *Journal of the Korean Information Technology Association*, vol. 19, no. 1, pp. 43–54, 2021.

[219] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, and L. Liang, "Astraea: Self-Balancing Federated Learning for Improving Classification Accuracy of Mobile Deep Learning Applications," in *37th International Conference on Computer Design (ICCD).* IEEE, 2019, pp. 246–254.

[220] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.

[221] B. Li, Y. Liu, and X. Wang, "Gradient Harmonized Single-Stage Detector," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8577–8584.

[222] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy, "Training Deep Neural Networks on Imbalanced Data Sets," in *International Joint Conference on Neural Networks (IJCNN).* IEEE, 2016, pp. 4368–4374.

[223] Z. Sun and H. Sun, "Stacked Denoising Autoencoder with Density-Grid based Clustering Method for Detecting Outlier of Wind Turbine Components," *IEEE Access*, vol. 7, pp. 13 078–13 091, 2019.

[224] G. P. Drago and S. Ridella, "Statistically Controlled Activation Weight Initialization (SCAWI)," *IEEE Transactions on Neural Networks*, vol. 3, no. 4, pp. 627–631, 1992.

[225] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[226] N. Drakoln, *Winning the Trading Game: Why 95% of Traders Lose and What you Must do to Win.* John Wiley & Sons, 2008, vol. 322.

[227] L. Le, A. Patterson, and M. White, "Supervised Autoencoders: Improving Generalization Performance with Unsupervised Regularizers," *Advances in Neural Information Processing Systems*, vol. 31, pp. 107–117, 2018.

[228] S. Gao, Y. Zhang, K. Jia, J. Lu, and Y. Zhang, "Single Sample Face Recognition via Learning Deep Supervised Autoencoders," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2108–2118, 2015.

[229] Q. Hu, R. Zhang, and Y. Zhou, "Transfer Learning for Short-Term Wind Speed Prediction with Deep Neural Networks," *Renewable Energy*, vol. 85, pp. 83–95, 2016.

[230] A. Van Opbroek, H. C. Achterberg, M. W. Vernooij, and M. De Bruijne, "Transfer Learning for Image Segmentation by Combining Image Weighting and Kernel Learning," *IEEE Transactions on Medical Imaging*, vol. 38, no. 1, pp. 213–224, 2018.

[231] Z. A. Khan, T. Hussain, A. Ullah, S. Rho, M. Lee, and S. W. Baik, "Towards Efficient Electricity Forecasting in Residential and Commercial Buildings: A Novel Hybrid CNN with a LSTM-AE based Framework," *Sensors*, vol. 20, no. 5, p. 1399, 2020.