April 2022

Data-driven Design and Analysis of Next Generation Mobile Networks for Anomaly Detection and Signal Classification with Fast, Robust and Light Machine Learning

Muhammed Furkan Küçük
University of South Florida

Data-driven Design and Analysis of Next Generation Mobile Networks for Anomaly Detection and Signal Classification with Fast, Robust and Light Machine Learning

by

Muhammed Furkan Küçük

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Date of Approval:
April 20, 2022

Keywords: Deep Learning, Deep Clustering, Wireless Communications, Self-organizing Networks

Dedication

To my Family

Acknowledgments

I would like to give many thanks to the Republic of Turkey Ministry of National Education, who supplied this scholarship to support my graduate education life and gave me this opportunity to make my dream real. I also thank the University of South Florida at Electrical Engineering Department, which partly supports my research.

I would like to thank you, Dr. Nasir Ghani, Dr. Srinivas Katkoori, Dr. Ozsel Kilinc, Dr. Zhuo Lu, and Dr. Alfredo Weitzenfeld for serving on my committee. I will always appreciate all they have done, especially Dr. Ismail Uysal, for believing in me, providing me the opportunity and the means to achieve this level, and having the patience to develop my skills.

It has been a great privilege to be an RFID Research Lab group member. This group provides a perfect research environment with very valuable collaborations. I would like to thank my friends Ria Kanjilal, Rania Elashmawy, Dr.Mehmet Aktukmak, and Dr.Alla Abdella for their support as friends and productive discussions as colleagues.

A special feeling of gratitude to my loving parents, Ömer and Dönüş Küçük, whose endless support with words of encouragement and push for tenacity ring in my ears. My sister Büşra and brother Faruk have never left my side and are very special. I also dedicate this dissertation to my many friends who contribute valuable efforts through the process.

My deepest gratitude goes to my wife, Şeyma Nur Küçük, for her sacrifice, support, and patience during my study. I could have never completed this dissertation without her. I hope our dear daughter Ayşe Hümeyra Küçük and my son Musab Bera Küçük always appreciate their mother and will achieve better accomplishments in their life.

Table of Contents

List of Tables

List of Figures

Abstract

This research focuses on machine (and deep) learning applications (including clustering, anomaly detection and signal classification) for self-organizing and next generation mobile networks in wireless communications. Specifically, this dissertation document will address the three different topics.

First, in the study titled "Performance analysis of neural network topologies and hyperparameters for deep clustering", we explore the relationship between the clustering performance and network complexity. Deep learning found its initial footing in supervised applications such as image and voice recognition successes of which were followed by deep generative models across similar domains. In recent years, researchers have proposed creative learning representations to utilize the unparalleled generalization capabilities of such structures for unsupervised applications commonly called deep clustering. This paper presents a comprehensive analysis of popular deep clustering architectures including deep autoencoders and convolutional autoencoders to study how network topology, hyperparameters and clustering coefficients impact accuracy. Three popular benchmark datasets are used including MNIST, CIFAR10 and SVHN to ensure data independent results. In total, 20 different pairings of topologies and clustering coefficients are used for both the standard and convolutional autoencoder architectures across three different datasets for a joint analysis of 120 unique combinations with sufficient repetitive testing for statistical significance. The results suggest that there is a general optimum when it comes to choosing the coding layer (latent dimension) size which is correlated to an extent with the complexity of the dataset. Moreover, for image datasets, when color makes a meaningful contribution to the identity of the observation, it also helps improve the subsequent deep clustering performance.

Second, in the study titled “Anomaly Detection in Self-Organizing Networks Conventional vs. Contemporary Machine Learning”, we compare the premise of both conventional and modern machine (deep) learning, specifically for anomaly detection in self-organizing networks. While deep learning has gained significant traction, especially in application scenarios where large volumes of data can be collected and processed, more conventional methods may yet offer strong statistical alternatives, especially when using proper learning representations. For instance, support vector machines have previously demonstrated state-of-the-art potential in many binary classification applications and can be further exploited with different representations, such as one-class learning and data augmentation. We demonstrate for the first time, on a previously published and publicly available dataset, that conventional machine learning can outperform the previous state-of-the-art using deep learning by 15% on average across four different application scenarios. Our results indicate that when execution time is critical, conventional machine learning provides a strong alternative for 5G self-organizing networks using significantly fewer trainable parameters.

Finally, the third study is on “Fast, Robust and Light Machine Learning for Signal Classification in Next Generation Mobile Networks”. The next generation mobile networks bring unprecedented opportunities coupled with unique challenges thanks to the integration of multiple families of devices. Fast and robust signal classification and modulation identification become critical to meet the sustained demand on capacity. This paper presents a comparative study of data-centric and conventional approaches to signal identification at different noise levels on a real-world application. We demonstrate that a standard lightweight classifier can detect multiple modulation schemes with and without data compression and outperforms current state-of-the-art by as much as 6% on average across 15 different noise levels. More importantly, the detection speed is improved by at least 50-fold without a significant loss in accuracy when using feature compression.

Chapter 1: Introduction and Contributions

The primary motivation of this dissertation is to introduce machine learning techniques to improve the performance of the next-generation mobile networks. The essential background and the main contributions of this research are detailed in the following sections along with the contributions of this thesis. Further details of the contributions can be found in the respective chapters.[1]

1.1 Performance Analysis of Neural Network Topologies and Hyperparameters for Deep Clustering

Clustering [102] is a pervasive problem in various research fields such as data analysis, machine learning, deep learning, computer vision, pattern recognition, signal identification, etc. The main goal is to group data into clusters based on the similarities or dissimilarities of the sample points. Most commonly used conventional clustering methods are applied to data with small volumes. However, in the era of deep learning and big data with multimodal and extremely high volume datasets the performance of the conventional clustering methods for identifying similarities are inherently weak. Dimensionality reduction techniques have been applied to raw data to lessen the volume by keeping the main characteristics of the sample features and allow conventional algorithms to achieve acceptable performance levels. In recent years, deep clustering had played a very important role where the power of deep neural networks (DNNs) and autoencoders [64] are used as feature extractors. Like conventional dimensionality reduction methods, deep neural networks can be used to transform

[1]Part of this chapter was published in [56]. The permission is includes in Appendix A

the raw data into a latent low-dimensional representation which can be directly applied to conventional approaches.

Categorization of clustering methods is based on their performance criteria including partition-based clustering methods, density-based clustering methods, and hierarchical clustering methods. Deep clustering first aims to learn a clustering-focused representation. For that reason, categorization of the deep neural network-based methods cannot be done solely on the clustering loss and instead should focus on the hyperparameters such as the clustering coefficient, network topology, and latent space representation (code) size.

The first chapter of this dissertation will focus on analyzing the effect of hypermeters on deep clustering (DC) performances to close the gap in our knowledge on how DC hyperparameters affect the clustering performances of deep embedding networks. The deep neural networks used in this study are auto-encoders (AE) [90], and convolutional auto-encoders (CAE) [39] applied to different image datasets with varying complexities. For drawing a clearer perspective, popular image benchmark datasets, which are MNIST, CIFAR10, and SVHN, have been chosen for a two-phase experiment where each phase implements a change in either topology or hyperparameter set for an exhaustive analysis of the field. The details are provided in chapter 3.

1.1.1 Contributions to Science

In chapter 3 we present novel approaches for analyzing clustering method performances by looking at the hyperparameters while applying the deep neural network-based techniques such as autoencoder and convolutional autoencoder on datasets with varying complexity. The contributions of this study are the findings that i) when using deep unsupervised feature extraction, more complex datasets require a higher dimensional latent space to achieve the best subsequent clustering performance and ii) unlike previous observations, the color information could be helpful in statistically significantly improving the deep clustering performance for datasets only when color makes a meaningful contribution to the identity of the observation.

1.2 Anomaly Detection in Self-Organizing Networks: Conventional vs. Contemporary Machine Learning

The second and third parts of this dissertation will focus on anomaly detection and signal classification in the next generation mobile networks. We begin with self-organizing networks (SON) and compare the detection rates of classical machine learning techniques and modern approaches such as deep learning for both complexity and performance to guide their implementation in the next generation mobile networks.

As mobile traffic data grew exponentially, hyper-connectivity, and various applications have increased the attention on new technology for the next-generation mobile networks. 5G is the latest cellular communications technology and it works to meet the demanding requirements of mobile customers (supporting a wider range of services and fulfilling the needs of new applications, etc.), which cannot be sustained by the current 4G technology [32]. It is vitally important that these new types of services supporting a divergent key technology embraced by 4G are required to properly integrate into 5G and beyond with additional development as necessary for future technologies. Machine learning and big data have created big opportunities to enable intelligent SON operations by using the data analysis process in the network for data driven decision making to enable the autonomous network management by fixing issues and optimizing operations.

Nowadays, telecommunication companies have updated their infrastructure according to the most recent technological developments such as 5G and beyond. Future systems will only bring new requirements based the business and customer needs. Specifically, artificial intelligence is one of the core requirements of future mobile network systems and must be deployed effectively and efficiently on the current systems. For that reason, the telecommunications companies have already begun to develop methods to support smart technologies by improving their infrastructures and adopting AI-related technologies.

Vodafone [101] is one telecommunications company with a significant analytical vision and strategies on AI adoption to provide network and cost efficiency in the radio domain and reduce its operational costs. Vodafone uses cloud economics by migrating to the Google Could Platform (GCP) to form a single data-ocean in collecting data – including the information related to network issues and service faults for a complete automatic network management strategy. For that reason, Vodafone cooperated with Nokia to develop AI/ML-based intelligent applications by using anomaly detection as a use-case while considering its potential for network automation. The intuition behind using anomaly detection is based on Vodafone's strategy on future cost-effective network planning and optimization while expanding coverage areas and minimizing risk. Vodafone deployed the general system structure with partnerships. In that structure, the Neuron platforms run with GCP to deliver and collect the Radio Access Network information via the platform by utilizing ML-based pattern recognition, clustering, and classification. Nokia helped Vodafone by developing an app to detect anomalies within the data provided by Vodafone called the data-ocean before an actual impact on the customers. The systems have already been deployed in Italy over *60000 LTE cells aiming for at least 80% productivity for all abnormal mobile network issues while automatically identifying the capacity demands by the anomaly detection service system.

SONs are considered automatic management systems for the next-generation wireless networks (NGWN) standardized by 3GPP [40]. The notion of SONs started with the eighth release of the 3GPP and continues through NGWN standardization based on self-configuration, building, and setting up and running of equipment. The SON aims to obtain targeted performance values and key performance indicators (KPIs). The KPIs can be evaluated within the telecommunication network's scope, including network capacity, quality of service (QoS), and capital and operational expenditures (CAPEX&OPEX). CAPEX refers to gaining and adapting new technologies and network assets, whereas OPEX relates to cellular networks' actuating operation and maintenance costs. Here, SON targets minimum CAPEX&OPEX by cutting human input as much as possible in network operations while optimizing the network coverage, capacity, and quality of service. Generally, SONs are evaluated in three categories: self-configuration, self-optimization, and self-healing. In this study we mainly focus on the self-healing aspect of SONs [7]. The self-healing can detect outages (anomalies) in cellular networks automatically. In addition to this capability, it provides performance degradation and root-cause analysis of cells, and compensation of outage affected cells if the issue is being resolved. Furthermore, it can also reduce operational costs by minimizing network outages and increasing the quality of service in an automatic manner.

Anomaly detection is a well-researched common problem [53] in various domains. It is generally referred to as identifying key differences between normal and anomaly samples. For instance, in wireless communications, the transmission of healthy knowledge in a proper channel is an expected norm. However, when the transmission process is not successful and the information does not reach the destination point an anomaly is flagged. The error could be hardware malfunctions, software problems, functional resource failures, loss due to overload situations, broken base stations, cell outage, etc. Anything that prevents communication is an unexpected "feature" and considered an anomaly in a network which can be detected by a variety of anomaly detection algorithms.

Autonomous separation of abnormal (anomaly) samples from the normal ones can be supplied via historical datasets labeled by the network operators. In those datasets, KPIs are manually classified as anomaly or normal, which helps train automated algorithms [106]. Unfortunately, this assumption cannot be generalized because the historical data may not contain a sufficient number of anomaly samples properly detected by the network operators. Hence, many research studies look into adding synthetic anomalies to obtain a dataset including a sufficient number of both sample types. Other approaches include no-label assumptions where each sample is treated as "normal" when the incidence rate of anomaly samples is too low to affect training.

In this study, the MDT (Minimization of Drive Test) report-based dataset generated in a network simulation [9] is used for training purposes. 3GPP first introduced the MDT reporting schemes in release 10, a standardized solution offered by network operators to reduce the cost of the conventional drive tests. The release commits to building a database of MDT reports from the deployed network using Immediate or Logged MDT information forms. The reporting is considered for the collection of each user equipment attitude in the network. The MDT report includes measurement information of UEs (User Equipments) as KPIs, which are defined as use cases. The KPIs consist mainly of Reference Signal Received Power (RSRP) and Reference Signal Received Quality (RSRQ). In this study we propose an alternative to the popular deep learning approaches when applying anomaly detection on a moderately sized labeled dataset where the details are provided in chapter 4.

1.2.1 Contributions to Science

In chapter 4, we introduce comprehensive analysis of classical and modern machine learning techniques in detecting anomalies (service outages) in self-organizing networks. This study has multiple contributions. Firstly, we present a comprehensive analysis of a conventional machine learning method for anomaly detection in self-organizing 5G networks (5G-SONs) and compare it with a popular deep learning alternative using different learning representations, including one-class and binary learning. We claim state-of-the-art performance on a publicly available dataset [9], which investigates multiple use case scenarios for anomaly detection in 5G-SONs where the results demonstrate an average improvement of 15% over the best recent performance which was achieved by a deep auto-encoder-based setup. Furthermore, we demonstrate for the first time that data augmentation methods can further boost anomaly detection performance in binary mode, even when utilizing conventional algorithmic techniques such as support vector machines on a sufficiently large dataset. Finally, we achieve nearly two orders of magnitude improvement in computational speed and an order of magnitude reduction in trainable parameters using conventional machine learning to provide a robust alternative for 5G self-organizing networks especially when the execution and detection times are critical.

1.3 Fast, Robust, and Light Machine Learning for Signal Classification in Next Generation Mobile Networks

The exponential growth of data transmission in wireless communications systems, millions of multi divergent additional devices and networks undeniably cause spectrum limitations in dynamic networks [93]. Scientists have put significant effort to respond to such demands with multidisciplinary research on utilizing small cells, mmWave communications, device-to-device communications, and massive MIMO for cognitive radio networks. Furthermore, authorities such as FCC, highlighted the importance of spectrum sensing algorithms, specifically signal recognition, to improve the quality of service by considering their effectiveness on various devices, network structures, transmission layout, and transmission channel conditions. Machine learning can provide the necessary intelligence function for wireless infrastructures to identify and detect the radio frequency (RF) signal characteristics implemented for commercial and military applications.

Electronic warfare and spectrum awareness have been employed to identify (classification) wireless signals in various tasks. Although the signal classification tasks rely on high order cumulants and hierarchical decision trees for feature extraction, they are susceptible to wrong alarms and their adaptations to new technologies can be difficult. Hence, data-driven machine learning-based approaches that include possible faults for real-world conditions can prove more precise, efficient, and reliable than classical methods. The signal identification methods are divided mainly into feature-based and likelihood-based methods [25]. Most studies that use the feature based methods employ cyclostationarity based-features for the most robust performance in real-world scenarios to account for channel effects and model mismatches, e.g., phase, timing, and timing frequency offset. In this dissertation we consider a feature based approach with cyclostationary- features obtained from real-world measurements. The dataset includes most of the commercially used frequencies with different modulations to define a multiclass classification problem. A feed forward neural network is used with an without data compression to compare the performance of signal classification with the state of the art approach on this dataset. Chapter 5 explains in detail how the proposed method achieves better performance with a lighter algorithm that is two orders of magnitude faster.

1.3.1 Contributions to Science

In chapter 5, we investigate the promise of conventional machine learning for signal classification (identification) that is utilized on a very large real-life dataset, where the signal characteristics represent the most-used channel frequencies in communications. The main contributions of this study are as follows. Firstly, the proposed model is more straightforward than complex deep learning topologies, which previously claimed state-of-the-art for signal classification on this dataset. Simplicity is essential for time-critical applications such as modulation classification in this case where the proposed model is shown to reduce the the identification time of signals significantly. Second, we discovered that PCA dimensionality reduction techniques in this study have an extraordinary impact on reducing the identification time even further with minimal impact on performance. After PCA, feature samples include useful information with less volume. Finally, we demonstrate that this model has better classification accuracy results (average of 6%) with or without data compression than the state-of-the-art study while running 50x faster. Overall, the proposed methods clearly demonstrate that the conventional machine learning algorithms are still valuable tools even for very large datasets while having the significant advantages of being fast and light.

Chapter 2: A Brief Introduction to Machine Learning

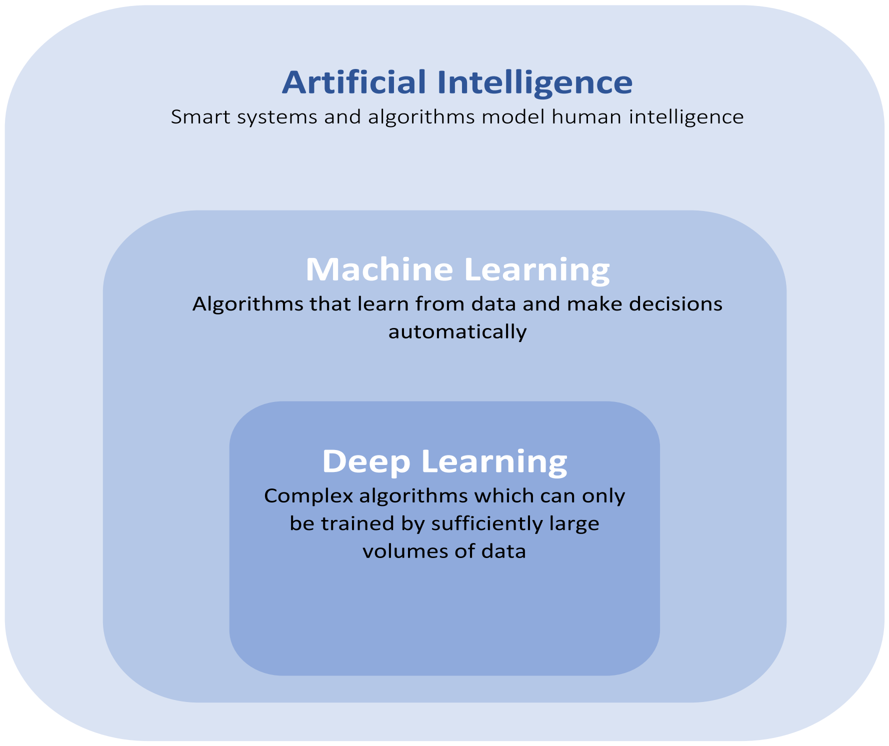

Artificial Intelligence
Smart systems and algorithms model human intelligence
Machine Learning
Algorithms that learn from data and make decisions automatically
Deep Learning
Complex algorithms which can only be trained by sufficiently large volumes of data

Figure 2.1: Comparison of Artificial Intelligence, Machine Learning, and Deep Learning

2.1 Learning Representations

There are mainly three techniques represented in Figure 2.1 which has been used in machine learning algorithms. Those are;

2.1.1 Supervised Learning

Supervised learning is one of the three main types of machine learning [85]. As the name suggests, it uses labeled datasets for training in many tasks such as classification or prediction. In supervised learning, classical machine learning training steps are applied which begin with feeding the input data to the model. The model updates the system weights until a convergence criteria is reached (i.e., the model fits to the data). Some kind of cross-validation can be used to prevent overfitting of the model to the data. Supervised learning has been used in many real-world applications, including text categorization, face detection, signature recognition, spam detection, weather forecasting, predicting house prices, etc. Popular methods for supervised learning include neural networks, naïve Bayes, linear regression, logistic regression, random forest, and support vector machines. In this dissertation, we used support vector machines and artificial neural networks.

2.1.2 Unsupervised Learning

Unsupervised learning is a training technique without the need for labels. Unsupervised learning primarily targets data clustering and dimensionality reduction problems which can further be used in a supervised training applications [100]. In the clustering applications the data is divided into distinctly separable groups based on the distances between data points based on some metric and generally within the latent space. Dimensionality reduction is commonly used to identify the most relevant features of the data and alleviate computational requirements which can come from high dimensional data. In unsupervised learning, the machine learning algorithms can automatically explore disguised information or data groupings. Popular methods include autoencoder neural networks, k-means clustering, probabilistic clustering, principal component analysis and singular value decomposition.

In this dissertation we utilize a novel learning representation called one-class training, to utilize supervised learning architectures in an unsupervised manner.

2.1.3 Semi-Supervised Learning

Unlike the previous two approaches, semi-supervised learning can utilize both labeled and unlabeled datasets in the training process [96]. Typically, the labeled data constitutes only a tiny amount of the training samples, whereas the unlabeled data represents most of the dataset. This technique has been based on self-training, also known as self-labeling/learning from the heuristic approach, the earliest one for semi-supervised learning with given initial examples in the 1960s. Later, it differed from transductive learning introduced by Vapnik in the 1970s along with the inductive learning for generative models in the same year. In the following decade, "probably approximately correct" learning was demonstrated by Leslie Valiant as a novel semi-supervised machine learning framework. Semi-supervised learning addresses the practical limitations of the data collection process while also decreasing the computational cost of training for a more practical approach.

2.1.4 Principal Component Analysis

Principal component analysis (PCA) is a dimensionality reduction technique [49],[50], also called a data compression algorithm that is often utilized to obtain low dimensional (uncorrelated variables) versions of large datasets (correlated variables) by still preserving the rich information on the original dataset. It is a popular method for compression due to the fact that it is non-parametric and straightforward, meaning that extracting relevant information does not follow a specific distribution in the dataset. In the literature, PCA has often been used for divergence fields. For example, in [1], it is used to extract the signal that has been subjected to noise or prevent the propagation effects in wireless communications. Another application in [66] proposes a PCA-based spectrum detection perspective for cognitive radio networks. If there is white noise in the signal, covariance matrix (CM) of the signal samples is diagonal. If the signal includes a component of the white noise, the CM becomes a low-rank diagonal matrix due to the fact that the primary signal is low rank. By subtracting the CM of white noise samples from the CM of signal samples and utilizing PCA on the remaining CM one can identify the significant principal components. After the PCA, the results can be used for test statistics for the spectrum sensing. Furthermore, in [105], the author has developed a PCA-base radio localization method. The method analyzes received signal strength samples and the user's location distribution to extract s location information.

Another definition of the PCA is an orthogonal linear transformation that represents the projection of observation (original variables) to a new coordinate system that maximizes the variance. The first principal component is the projection of data with the greatest variance. The second principal component lies on the second coordinate that is orthogonal to the first, and so on.

In this chapter we focus on the eigenvector decomposition [89] as the main mathematical intuition behind the PCA as follows:

Let X be the original data set, mxn, where each column represents a single observation. Let Y be another mxn matrix related to X by a linear transformation $P.X$ where Y becomes a new projection of that data set as follows:

$$P.X = Y \tag{2.1}$$

$$PX = \begin{bmatrix} p_1 \\ \vdots \\ p_m \end{bmatrix} \times \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \tag{2.2}$$

$$Y = \begin{bmatrix} p_1.x_1 & \dots & p_1.x_n \\ \vdots & \ddots & \dots \\ p_m.x_1 & \dots & p_m.x_n \end{bmatrix} \tag{2.3}$$

where p_i are rows of P, x_i are the columns of X, y_i is first columns of Y. Please note that, from the first row of Y, we recognize that rows of P are a new set of basis vectors for the projection of the columns of X.

Identifying the principal components can be derived using many approaches. We represent an algebraic solution to the PCA by using "eigenvector decomposition". From the definition above, m is the number of measurement types, and n is the number of samples.

The goal is finding an orthonormal matrix P in $Y = PX$, where $Cy = 1/nYY^T$ is a diagonal matrix and the rows are the principal components of X. The following equations present this process mathematically:

$$\begin{aligned} C_{\mathbf{Y}} &= \frac{1}{n}\mathbf{YY^T} \\ &= \frac{1}{n}(\mathbf{PX})(\mathbf{PX})^{\mathbf{T}} \\ &= \frac{1}{n}(\mathbf{PXX^TP^T} \\ &= \mathbf{P}(\frac{\mathbf{1}}{\mathbf{n}}\mathbf{XX^T})\mathbf{P^T} \\ C_Y &= \mathbf{PC_XP^T} \end{aligned} \tag{2.4}$$

where C_y is the covariance matrix of X and C_x is a square symmetric mxm matrix. The covariance matrix is nothing but a table that summarizes the correlations between all the possible pairs of variables in a matrix. The positive sign of covariance shows that there is a positive correlation in between variables whereas the negative sign means inverse correlation.

The diagonal terms of C_x are the variance of measurement types. The off-diagonal terms of C_x are the covariance between the different measurement types. The goal is to diagonalize C_y as follows:

$$\begin{aligned} C_{\mathbf{Y}} &= \mathbf{PC_XP^T} \\ &= \mathbf{P(E^TDE)P^T} \\ &= \mathbf{P(P^TDP)P^T} \\ &= \mathbf{(PP^T)D(PP^T)} \\ &= \mathbf{(PP^{-1})D(PP^{-1})} \\ C_Y &= \mathbf{D} \end{aligned} \tag{2.5}$$

where $A = E^T DE$ is a symmetric matrix, D is the diagonal matrix, and E is a matrix of eigenvectors of A.

2.2 Machine Learning Algorithms and Topologies

2.2.1 Perceptron

The perceptron [84, 12] can be considered the basic building block of an artificial neural network. It consists of a single linear threshold unit (LTU) (although a non-linear activation function can also be used) after each input is weighed and summed. Instead of binary on/off values, the input and output of the perceptron have real values. Each input is associated with a weight. The linear threshold unit computes a weighted sum of inputs

$$(z = \omega_1.x_1 + \omega_2.x_2 + \ldots + \omega_n x_n = \omega^T.x) \tag{2.6}$$

The weighted sums goes through step (activation) function and the output is

$$f\omega(x) = step(z) = step(\omega^T.x) \tag{2.7}$$

There are two types of common step functions for a perceptron: 'heaviside' and 'sign'.

$$H[z] = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases} \tag{2.8}$$

$$sgn(z) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{2.9}$$

A single perceptron can be used as a simple binary classifier. Depending on the activation function, the calculated linear combination goes through the threshold and predicts either

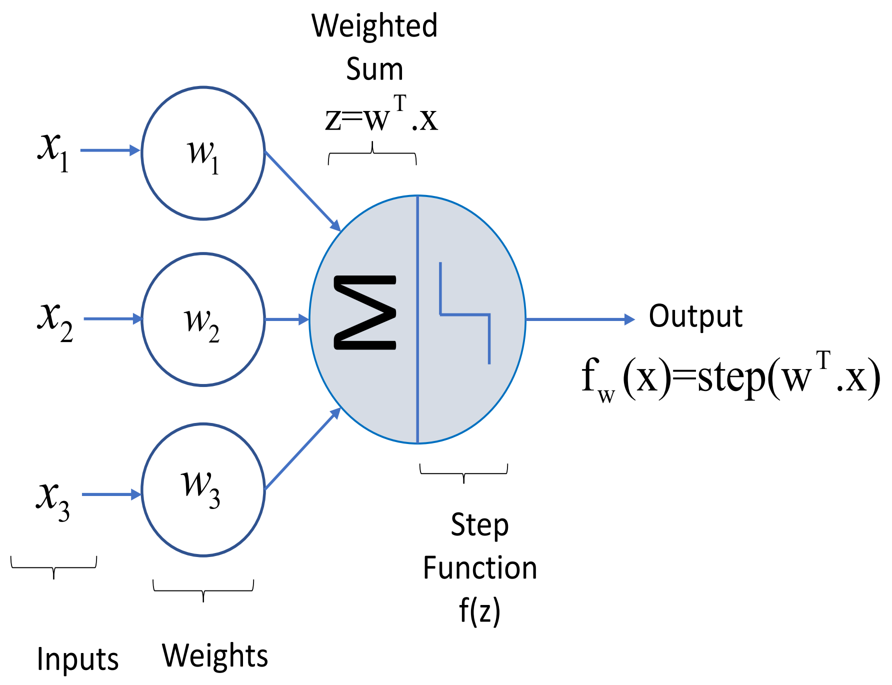


Figure 2.2: General View of Perceptron

a positive or negative class (prediction). During the training of the perception, the weight values for w0, w1, ... wn, etc. are updated to find the best model fit to the available data. The perceptron learning can be simplified mathematically as follows:

$$w_{i,j} = w_{i,j} + \theta(y_j - \hat{y}_j)x_i \tag{2.10}$$

where $w_{i,j}$ represents the connection weight between the i^{th} input neuron and the output neuron, x_i is the i^{th} input value, $\hat{y}_j$ is the output of j^{th} output neuron, y_j is target output, and θ is learning rate. Based on the eqn 2.10, the perceptron updates the weight parameters using gradient descent, however it is important to note that the single dimensional linearity prevents the perceptron from deciding on famous classification examples such as the exclusive OR gate.

2.2.2 Multilayer Perceptron

Multilayer perceptron can overcome the limitations of a single perceptron such as the decision boundaries for the XOR gate [71]. It consists of an input layer followed by one or multiple hidden layers. Every neuron except the ones in the output layer carries a bias neuron, and unlike the perceptron, each layer can use an arbitrary activation function.

Multilayer perceptrons are often called feedforward neural networks [34]. At the beginning of the training, each input sample is fed into the network where activation functions are calculated for each layer to be forwarded into the next layer progressively until the output layer. This one time pass is called the forward pass. Then, the sum of squared differences between the desired and original values of the output neuron is calculated as the error via the cost function shown below:

$$E = \frac{1}{2}\sum_{j}(y_j - \hat{y}_j)^2 \tag{2.11}$$

where y^{th} is the desired output of the j^{th} neuron in the output layer and y^{th} is the actual output. This error is backpropagated through the layers to find out the contribution of each neuron from the last hidden layer to the input layer in a process called backpropagation. Specifically, the backpropagation algorithm does a forward pass, calculates the error, then goes back through each layer to calculate the error contribution between neurons (a reverse pass). Eventually, it slightly updates the connection weights, $\triangle\omega_{ij}(n)$ to decrease the error, (E), via the gradient descent algorithm as shown below:

$$\triangle\omega_{ij}(n) = -\theta\frac{\partial E(n)}{\partial v_j(n)}y_i(n) \tag{2.12}$$

where y_i is the output of the i^{th} neuron (which can be considered as the input to the j^{th} neuron in the next layer), n is the index of the training example, v_j is the activation potential of the j^{th} neuron in the next layer, and θ is the learning rate, which is chosen to help increase the convergence speed for the weights. To calculate the degree of error rate at the output neuron, the partial derivate of eqn 2.12 can be simplified to:

$$-\frac{\partial E(n)}{\partial v_j(n)} = (y_j - y_j)\alpha'\hat{(}v_j(n) \tag{2.13}$$

where α' is derivative of the activation function. Changing the weights to the hidden layers is more complicated , but using the chain rule yields the following:

$$-\frac{\partial E(n)}{\partial v_j(n)} = \alpha'(v_j(n)) \sum_k -\frac{\partial E(n)}{\partial v_k(n)} \omega_{kj}(n) \tag{2.14}$$

where k is the neuron index in the previous layer going into the j^{th} neuron in the next layer.

In addition to the previous analysis, it is worth mentioning that the backpropagation algorithm can only be used actively in multilayer perceptron if the activation function is usable with the gradient descent algorithm (i.e., it's differentiable) [42]. Most commonly used logistic activation functions with backpropagation are summarized below.

$$S = \frac{1}{(1 + exp(-z))} \tag{2.15}$$

$$H = tanh(z) = 2\sigma(2z) - 1 \tag{2.16}$$

where S represents signum function, and H denoted "Hyperpolic Tangent" function.

2.2.3 Support Vector Machine

The support vector machine (SVM) is a family of machine learning algorithms [68]. It is an effective supervised learning method commonly used for regression and classification. It was developed and introduced by Vapnik at the ATT Bell Laboratories in 1997. SVM aims to discover the optimum region that separates two or more class members from each other.

The optimum area is often called a hyperplane if the classification problem is linear and the SVM does a linear separation.

On the other hand, if the classification problem cannot be solved with a single hyperplane, it requires non-linear classification. In this case, a kernel trick is applied for distinguishing the classes by mapping the non-linear feature data to higher dimensions where they become separable. The intuition behind the process for finding the best hyperplanes, both for linear and non-linear problems, are summarized as follows.

In linear problems [12], a given dataset $\{(x_1, y_1), ..., \} x_i \in R^n,\ y_i \in \{+1, -1\}$ can be separated by a hyperplane linearly. An n-dimensional vector w and a bias constant b, defines the hyperplane through the linear equation $w.xi + b = 0$. The perpendicular (normalized) distance from the hyperplane is found through the dot product where the hyperplane threshold function is used to make the following binary decision:

$$h(x_i) = \begin{cases} (w.x_i) + b > 0,\ y_i = 1 \\ (w.x_i) + b < 0,\ y_i = -1 \end{cases} \tag{2.17}$$

In the eqn 2.17, it is designated to belong to the positive class (+1) and vice versa for the negative class (-1). Thanks to this duality and the way the problem is formulated, the following equation is true for all the observations in the dataset:

$$y_i(\mathbf{W}^T X_i - b) \geq 1, \text{ for all } 1 \leq i \leq n \tag{2.18}$$

From the eqn 2.18, the goal is to find the maximal margin distances, geometrically defined as $\frac{2}{||\mathbf{w}||}$, between the planes that separate the two classes. The data points closest to the margins of the hyperplane become the so-called support vectors.

When the data is not linearly separable then the hinge loss function can replace the regular thresholding function as follows:

$$\max(0, 1 - y_i(w^T x_i - b)) \tag{2.19}$$

The hinge loss function is formulated such that it becomes zero if the condition in eqn (2.18) is satisfied. Also, the conditions are totally provided on the eqn (2.18) then following the non-linear problem be linear problem similar which is expressed as,

$$\lambda \|w\|^2 + \left[\frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T x_i - b)\right] \tag{2.20}$$

where λ (regularization parameter) is minimum and greater then zero to satisfy max margin size where the support vectors are correct side of the middle hyperplane (on figure it is shown as red). This process is also called as soft-margin and it is a technique for solving noisy data.

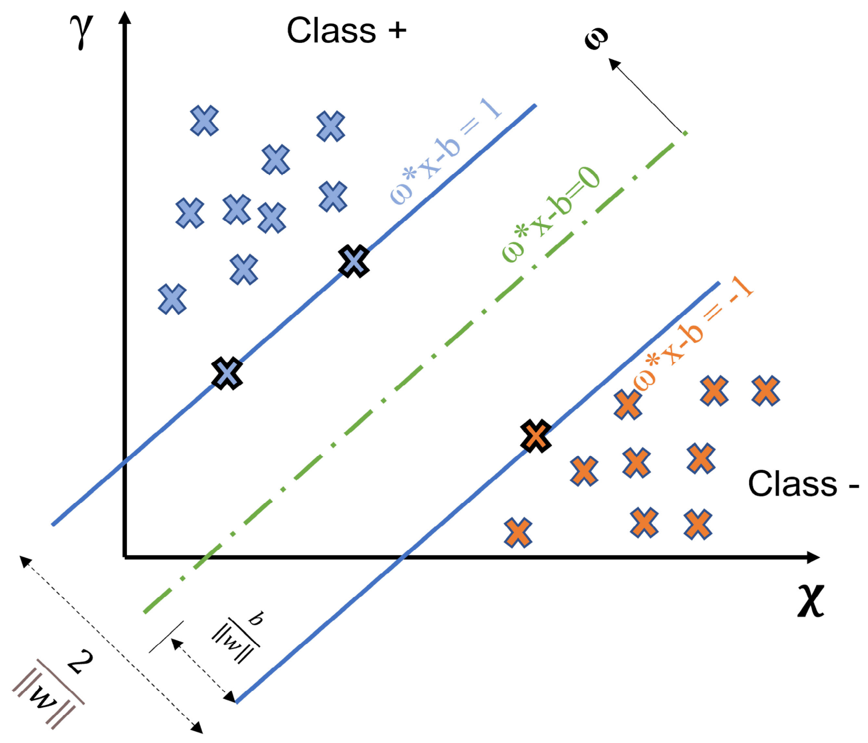


Figure 2.3: Support Vector Machine

When the data is not linearly separable one can apply a kernel trick where the data is projected into a higher dimensional space where it is separable. Another benefit of the kernel function is increasing the computational speed compared to the classical mathematical calculations especially for dot products. The kernel function is shown as $K(x_i, x_j) = x_i.x_j$. where polynomial and Gaussian kernels are two of the most popular forms.

$$K(x_{i,}x_j) = (x_i x_j + c)d \tag{2.21}$$

The eqn 2.21 is polynomial kernel. Here, c is a constant d represents the dimension of the projection space. As d As d increases, the decision boundary can become more complex and could cause overfitting. However, probably the most popular kernel shown to demonstrate superior performance in many non-linear applications is the radial basis function (RBF) kernel defined as follows:

$$K(x_i, x_j) = \exp(\lambda \|x_i x_j\|^2) \tag{2.22}$$

The egn 2.22 is also called the Gaussian kernel due to the shape of the kernel following the Gaussian distribution. Lambda is the hyperparameter of the kernel where smaller values generally yield better results.

2.3 A Brief History of Artificial Neural Networks

The concept of a "neural network" has been inspired by the fundamental operation of the human brain. The neural network uses "connectionism" described by the scientists as simulating the connected circuits to understand the intelligence of the human brain. In 1943, this concept was described with a single electrical circuit by neurophysiologist Waren McCulloch and mathematician Walter Pitts. This concept was further described in Donald Hebb's book, "The Organization of Behaviour (1949)", which talks about strengthening the neural network pathways for every single attempt as he famously put “the neurons that fire together wire together”.

Two main views are generally considered as pioneers to Neural Networks: “Threshold Logic” and “Hebbian Learning” which were both proposed in the 1940s. As researchers were trying to implement these networks in computational machines, a “Hebbian Network” was first implemented at MIT in 1954 called a ‘calculator’ at the time. In 1958, a psychologist at Cornell, Frank Rosenblat, proposed the idea of a “Perceptron (Mark I)” when he was trying to solve a different research problem. The general idea was modeled after Waren S. McCulloch’s and Walter Pitts’ 1943 McCulloch-Pitts neuron. The McCulloch-Pitts neuron explains a brain’s complex decision process using a simple linear threshold gate. The neuron takes inputs, applies a weighted sum, and returns ‘1’ if the result is above the threshold, and ‘0’ otherwise. This threshold model, began the history of the artificial neural networks (ANN).

Kelly in 1960 and Bryson in 1961, laid the foundations of “backpropagation” in control theory by using features of dynamic programming. In 1969, Minsky and Papert introduced the basics of perceptrons. However, the perceptrons did not satisfy expectations for solving trivial problems (e.g., the famous Exclusive OR or XOR problem). Hence, research stagnated until the elimination of basic perceptrons’ limitations by stacking multiple perceptrons together. It was called a Multi-Layer Perceptron (MLP) – the first neural network similar to its more recent cousins. There were significant developments in ANNs in the years followed by improving the hardware operations to provide robust and better alternatives for training to solve more complex problems.

In the 1990s, the concept of machine learning was introduced to the general field of AI with the help of MLPs in popular applications. Until the mid-1990s, the development of AI around the concept of neural networks was slow since it couldn’t satisfy mostly unrealistic expectations. However, at the same time, other branches of machine learning, such as kernel machines (Boser et al. 1992, Cortes and Vapnik, 1995; Scholkopf et al., 1999) and graphical models (Jordan, 1998) achieved satisfactory performance in a variety of tasks. These two methods continued to be more popular than neural networks until late 2000s.

Hochreiter (1991) and Bengio et al. (1994) have identified some mathematical difficulties in modeling sequences with neural networks. Later, Hochreiter and Schmidhuber (1997) presented long-term, short-term memory (LTSM) networks to solve these mathematical challenges through a novel topology which uses input, output and forget gates to capture the most relevant information in a sequence of data points such as speech signals. In today's world, LSTM is used in many sequence-based modeling tasks by large companies such as Google for natural language processing. A breakthrough in 2006 by Geoffrey Hinton and his colleagues introduced the greedy layer-wise pretraining to provide more efficient neural network training. Later, the same idea was successfully applied to other deep neural networks by Bengio in 2007 (Ranzato 2007). The term "deep learning" became popular among the scientists as more applications began to outperform conventional machine learning and feature extracting approaches in fields from image processing to speech(Bengio and Lecun, 2007; Delalleau and Bengio, 2011; Pascanu et al., 2014a; Montufar et al., 2014).

2.3.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) were presented in the early 1980s by Yann LeCun [58] with inspiration from the previous work done by the Japanese scientist Kunihiko Fukushima, who invented the neocognitron, a simple image recognition network. Later, the CNN structure called LeNet [57] has gained the ability to recognize handwritten digits and became the most used tool in the banking and postal services. Their purpose was to read numbers and letters, especially the zip codes on arrival/destination documents and digits on important papers such as checks. Although the convolutional neural network was an intelligent tool, it lacked scalability as the CNNs required vast amounts of data for effective work on larger images. Hence, they were applied to only low-resolution images at the beginning. In 2012, Alex Krizhevsky won the ImageNet computer vision award by introducing AlexNet

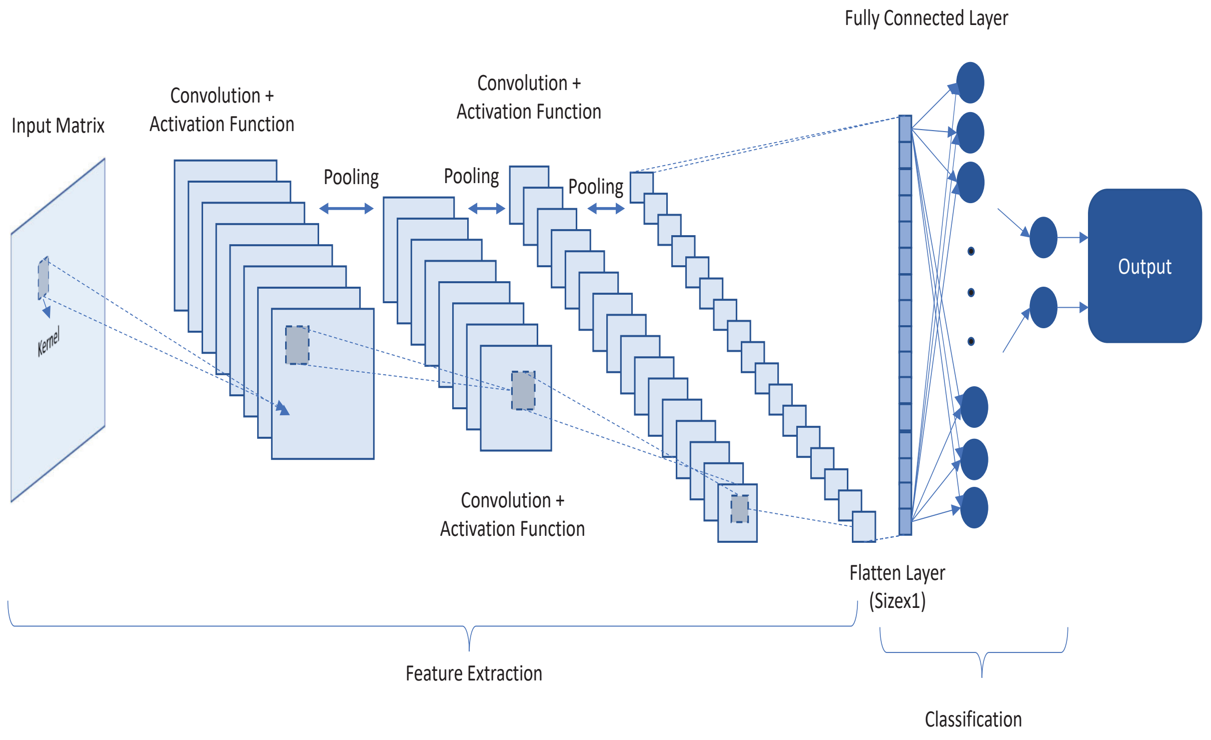


Figure 2.4: Convolutional Neural Network Structure

[55] which provided unprecedented 85% training accuracy and 74% test accuracy. AlexNet became the first CNN that could be applied to complex tasks by processing a high volume of image datasets. CNNs are now commonly used in computer vision, media recreation, recommendation systems, and natural language processing for text classification, etc.

The convolutional neural network is a branch of deep learning [38]. It functions partially as a feed-forward neural network with a slight difference. The first hidden layers are replaced with convolutional layers, followed by activation layers that are also called feature maps, and pooling layers for effective feature selection. The name comes from the convolution operation in the first few layers. As the convolutional layers increased in size thanks to widely available computational powers of graphical processing units or GPUs, the CNN's recognition capabilities also increased. For instance, while the handwritten digits could be recognized with just three or four layers, the human face needs more than twenty layers to effectively capture the fine details. The human visual cortex inspired the intuition behind the utilization of convolutional layers. The sequenced convolutional layers regularly process the images whenever they arrive at the next stage and resolve complex information as features. During this filtering process specific tasks are carried out such as padding, stride, convolution, and pooling.

Filtering is an operation to extract specific features from the input image. During the forward propagation (forward pass), each filter is convolved with the input matrix – across both the width and height of the samples of images to obtain a 2-D feature (activation) map. In other words, the convolutional network learns to identify basic information such as sharp edges and corners from the input matrix via trainable filtering operations. Stride and padding are important operations during the filtering. Stride controls how much sliding should be over the entire input matrix. For instance, if the stride is 1, the filter moves only for one pixel at a time.

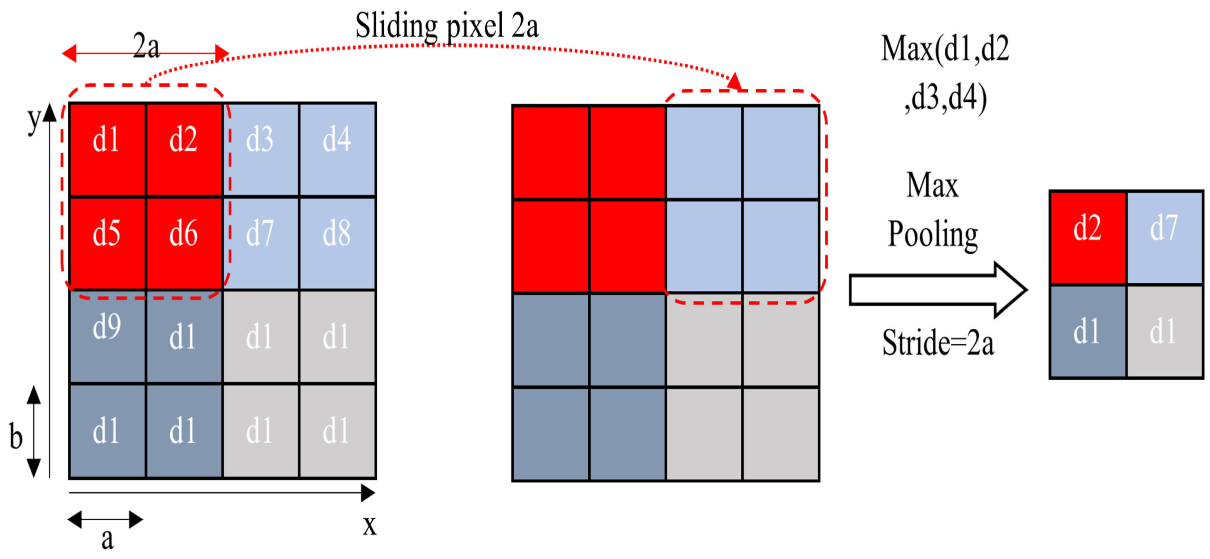


Figure 2.5: Filtering Operation in the CNN

Padding is one of two options to make the edges of the input matrix useful by adding information such as zeros which is called zero padding. The padding operation is required when the filter size (2a+2b) does not match exactly with the image dimensions to extract all the information from the input matrix. It also depends on the stride number from the edge

edge of the input matrix. That means, during the convolutional operation, movement of the filter on the x (horizontal) and y (vertical) axes, the filter covers the vast amount of information on the inner area of the input matrix wheras the edges would be less convolved. The other option is removing the parts of the input matrix where the filter does not fit. This option is called valid padding, which keeps only the necessary pieces of the input and removes the others.

Another important step in the filtering process is pooling. It is a non-linear down-sampling layer in the convolutional neural network. It divides the input matrix into sub-categories according to the filter size and takes the most useful information. That means it progressively reduces the input matrix size (fewer parameters) and controls overfitting. The pooling layer comes after the activation function. There are many pooling layers, but the most commonly used one is max-pooling. As the name suggests, it takes the maximum value from each filtered sub-part of the input matrix as output as shown:

$$f_{X,Y}(S) = \max_{a,b}(S_{2a+2b}) \tag{2.23}$$

where S is the size stride, and a and b are width and height of the filter.

Defining the network parameters for CNN to measure the complexity of this deep neural network is an important concept [4]. There are specific hyperparameters to choose including the output size of feature maps; the amount of padding and the number of trainable parameters within the particular layer.

The size of the output feature map, M, is calculated as follows:

$$M = \frac{(N - F)}{S} + 1 \tag{2.24}$$

where N refers to the volume of the input matrix, F represents the dimension of the filters, and S is defined as the stride number or length.

In addition, the amount of padding, P, is formulated as:

$$P = (F - 1)/2 \tag{2.25}$$

whereas F denotes the kernel size. Finding the number of parameters and the total memory required to process a specific (lth) layer are as follows;

$$N_p^{(th)} = (Fx(F+1)xFM_{l-1})xFM_l \tag{2.26}$$

where N_p^{lth} represents the total number of parameters of the l^{th} layer. FM_l is the total number of output feature maps and $FM_{(l-1)}$ is the size of the input matrix goin into the l^{th} layer.

The overall operation and structure of the CNN are shown in Figure 2.5. The red dotted box represents the sliding window as the filter moves based on the stride number. Output (activation map) is the compression of both the information and volume of the input matrix after the convolution operation. Pooling operation takes the maximum value, such as $d2$, among the calculated values $(d1, d2, d3, d4)$. The other values $(d7, d10, d15)$ are obtained similarly.

2.3.2 Convolutional Autoencoders

The convolutional autoencoder (CAE) [10] represents the joint operation of the autoencoder and convolutional neural network topologies commonly used to represent 2D images in the latent space. It has mainly two parts including encoding, and decoding layers. The encoding layers are similar to the CNN forward propagation structure and consists of convolution and max-pooling layers (downsampling). On the other hand, the decoder layers include deconvolution and upsampling (upscaling) layers. As we have seen from the autoencoder, the input data is compressed through the convolution and pooling operations in the encoder to be represented in the latent space. Then, the decoder part uses the compressed matrix in the latent space by applying the deconvolution and upsampling process to obtain

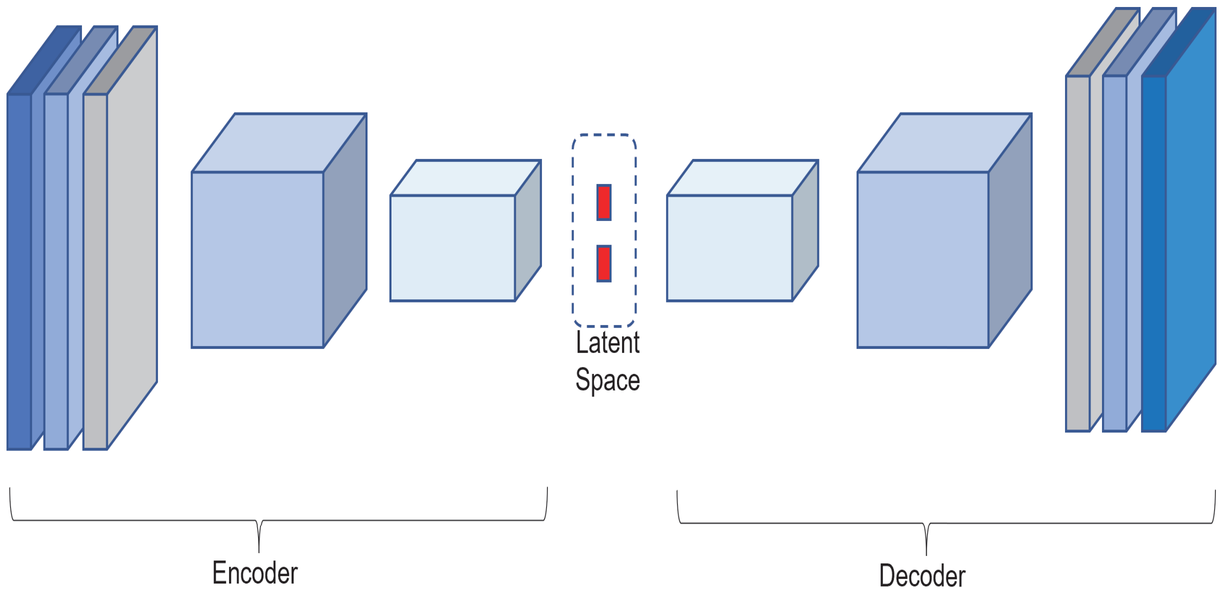


Figure 2.6: Convolutional Autoencoder Structure

the output matrix which aims to recreate the input matrix. The aforementioned process is illustrated in figure 2 and could be defined mathematically as follows:

The compressed features of the input image data, D with dimensions $K = K_1, K_2, \ldots, K_d$, n are obtained by passing through n convolution filters $F^{(1)} = F_1^{(1)}, F_2^{(1)}, \ldots, F_n^{(1)}$ to create intermediate (latent) features as follows:

$$T_l = f(K * F^{(1)_l + b_l^{(1)}}), m = 1, 2, ..., n \tag{2.27}$$

where f denoted the activation function such as relu or sigmoid, $b_l^{(1)}$ is the bias for l^{th} feature map.

During the decoding process, the reconstructed input image is represented as $\tilde{K}$, which is obtained by deconvolutional operationsas follows:

$$\widetilde{K} = f(T * F_l^{(2)} + b_l^{(2)}) \tag{2.28}$$

where $T = (T_l)_{l=1}^{n}$ denotes the intermediate (latent) feature maps, $F^{(2)} = F_1^{(2)}, F_2^{(2)}, \dots, F_n^{(2)}$ represent n deconvolutional filters for the decoding part. A loss function, L such as the mean squared error could then be used to update the weights in the network using backpropagation as explained in the previous chapters until the loss function reaches a threshold.

$$L(K, \tilde{K}) = \frac{1}{2}\left\|K - \tilde{K}\right\|_2^2 \tag{2.29}$$

That means the output features approximate the input features using only the transformation of the latent space.

Chapter 3: Performance Analysis of Neural Network Topologies and Hyperparameters for Deep Clustering

3.1 Introduction

Clustering [46], the unsupervised process that groups similar data examples together based on some distance measures, is one of the primary problems in various research fields, such as machine learning, computer vision, pattern recognition and, data analysis. Many clustering methods have been proposed including k-means [6],[41],[99] and Gaussian Mixture Models (GMM)[81]-[12], however, traditional clustering methods do not perform well with high-dimensional data, due to the inadequacy of distance measures applied in these methods. Besides, these clustering methods are affected by high computational complexity on large datasets. Therefore, dimensionality reduction and feature mapping methods have been studied extensively to represent the original data in a feature (latent) space where original data is separated more effectively by a clustering algorithm. However, the complexity of the latent space still remains a challenging problem. Recent progress in deep learning [88], led to deep neural networks (DNN) being used as non-linear and rich mappings of the data input space into a lower dimensional feature space. In other words, DNNs integrate representation learning with clustering using raw data with a high accuracy rate. This new method of grouping is generally referred to as Deep Clustering (DC).[2]

Researchers have previously considered feature mapping and data grouping (clustering) as two different processes. First, high dimensional input examples are transformed into a generally lower dimensional feature space. Then, the clustering algorithm is applied to the transformed data. DC on the other hand aims to combine these two processes as first intro-

[2]Part of this chapter was published in [56]. The permission is includes in Appendix A

duced with the Deep Embedding Clustering (DEC) [3] which implements feature mapping via a fully connected deep auto-encoder [98] with a k-means back-end for clustering. Variations of DEC have been proposed in recent years including, the Discriminatively Boosted Clustering (DBC) which replaces the feature mapping auto-encoder with a convolutional auto-encoder (CAE) for image analysis [60], a joint dimensionality reduction technique with k-means based on DNN [103], the Deep Embedded Regularized Clustering (DEPICT) using logistic regression with CAE for joint clustering assignment [37], the Variational Deep Embedding (VaDE) based on a variational auto-encoder (VAE) and Gaussian Mixture Model (GMM) [47],the Joint Unsupervised Learning (JULE) proposed as a recurrent perspective with convolutional neural network (CNN) activated data on agglomerative clustering [104], and a CNN-based joint clustering method which brings an iterative solution with feature drift compensation [43]. While deep clustering remains a popular research field with such recent advances in algorithm design and clustering accuracy [73], the process of choosing many of the hyper - parameters, such as the code size, network topology and clustering coefficient, still remains an inexact science.

The purpose of this analysis study is to address this gap in our knowledge of deep clustering methodologies and conduct a comprehensive analysis study on how DC hyperparameters affect the clustering performances of deep embedding networks. The DNNs used in this study are auto-encoders (AE) [90] and convolutional auto-encoders (CAE) [39] applied to different image datasets with varying complexities. For a clear perspective, we choose popular image benchmark datasets MNIST, CIFAR10, and SVHN for a two-phase experiment where each phase implements a change in either topology or hyper-parameter set. In summary this study has the following contributions:

1. When using deep unsupervised feature extraction, more complex datasets require a higher dimensional latent space to achieve the best subsequent clustering performance.

2. Color information can be useful in statistically significantly improving the deep clustering performance for datasets where color makes a meaningful contribution to the identity of the observation.

3. General trends need to be further evaluated on a wider variety of datasets and clustering domains to make more definitive conclusions.

The rest of this chapter is organized as follows: section 3.2 provides general information about the Autoencoder and the Convolutional Autoencoder structures and their applications to deep clustering. Section 3.3 describes the experimental setup used in the study and along with the hyperparameters, the datasets, evaluation metrics, and implementation. The results and discussions are presented in section 3.4.

3.2 Methods

3.2.1 Autoencoder

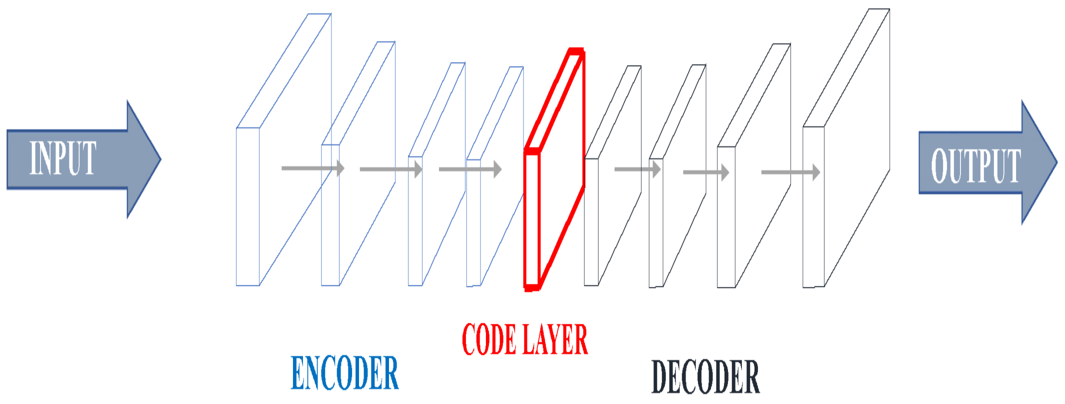


Figure 3.1: Autoencoder Network Structure

Autoencoder is a particular artificial neural network topology which has the same input and output layers where the training is performed by presenting the same input data to both layers simultaneously. The general structure of the auto-encoder consists of a visible input layer $\mathbf{x}$, a number of hidden layers $\mathbf{h}$ and the reconstructed output layer $\mathbf{y}$ with a family of nonlinear activation functions $\mathbf{f}$ applied at different layers.

During training, the auto-encoder maps the input $\mathbf{x} \varepsilon \mathbf{R^y}$ to the hidden layers with lesser dimensions than the input data which produces a compressed representation of the original data in which its dimensionality is reduced to the code (latent) layer size $\mathbf{H} \varepsilon \mathbf{R^h}$. This first step is called the "encoder" and is shown on the left side of Figure 3.1. Later, the compressed information is mapped to the output layer via the "decoder," through a process called "reconstruction." Mathematically, these two steps are formulated as follows:

$$H \equiv f_{WH}(x) = f(W_H x + b_H) \tag{3.1}$$

$$z \equiv g_{W_z}(x) = g(W_z H + b_Z) \tag{3.2}$$

where W_H and W_Z define the encoding weight and decoding weight, respectively, b_H and b_Z define the corresponding encoding bias vector and the decoding bias vector, and $f(.)$ and $g(.)$ are encoding and decoding activation functions such as a sigmoid function or a rectified linear unit, respectively. As previously mentioned, the primary purpose of the auto-encoder is to learn useful latent information on the code layer by minimizing the reconstruction error. For a given N input data samples, the following loss function is used to determine the parameters "W_H, W_Z, b_H, and b_Z" through a back-propagation algorithm commonly used in feed-forward neural networks:

$$L_{AE} = min \frac{1}{N} \sum_{k=1}^{N} ||x_k - z_k||^2 \tag{3.3}$$

In this study, we constructed several auto-encoder networks with different topologies and four different code layer sizes to simulate a variety of scenarios and study the impact of topology on reconstruction and clustering performance.

3.2.2 Convolutional Autoencoder (CAE)

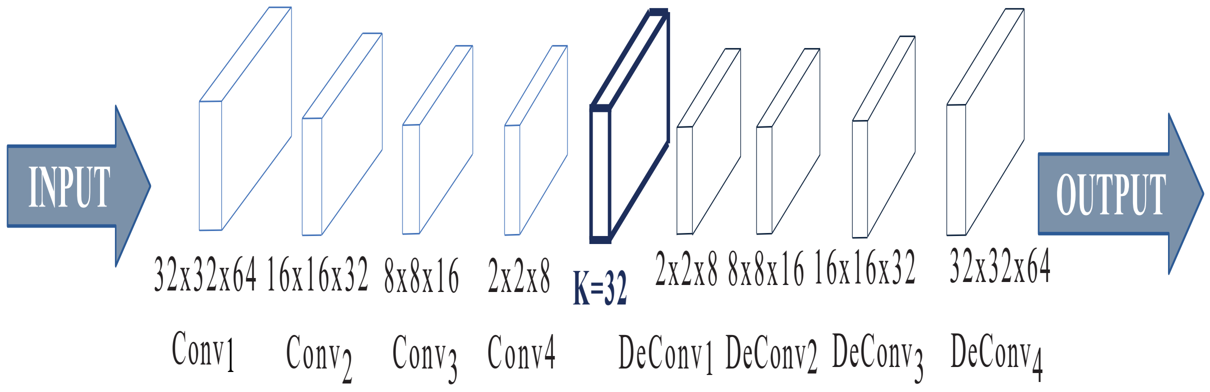


Figure 3.2: Convolutional Autoencoder Network Structure

The convolutional auto-encoder (CAE) is similar to the standard auto-encoder except the input layers are replaced with convolutional layers to present a powerful technique specifically for image-processing tasks. CAEs borrow ideas from the Convolutional Neural Networks (CNN) much like how AEs implement standard fully - connected networks. Similar to the equations defined in section II-A, we define the CAE encoding part as follows where multiplications are replaced with 2D convolutions:

$$H \equiv f(W_H * x) \tag{3.4}$$

$$z \equiv g_{DZ}(H) = g(H * D_Z) \tag{3.5}$$

where H represents the input image samples as the latent variables in the code layer which then feeds into the fully connected AE hidden layers, W_H and W_Z are encoding and decoding weights, '*' is the 2D convolution operation. The CAE's primary purpose is finding the latent layer representation, sometimes called the coding layer, through minimizing a cost function such as the mean squared error (MSE) between original and reconstructed images where the corresponding loss function is defined as:

$$L_{CAE} = \min_{W_H, D_Z} \frac{1}{N} \sum_{j=1}^{N} ||g_{D_Z}(f_{WH}(x_j)) - x_j||^2 \tag{3.6}$$

where N is the number of input images in the dataset, $x_j \varepsilon R^2$ is the j^{th} image.

As shown in Figure 3.2, each convolutional layer at the encoder includes filters with a certain size and stride, image normalization followed by max pooling to transform and compress the information included in the original image. The decoder structure is similar but in reverse order which includes up sampling to obtain the reconstructed image at the output layer of the autoencoder.

3.2.3 K-means Clustering

Clustering is performed on unlabeled observations in a dataset with the objective to group similar data samples in an unsupervised fashion. One of the most popular clustering algorithms is k-means which stands out from others with the guaranteed convergence property [63]. The hyper-parameter k defines the number of randomly assigned centroids which would be used to identify the center location for each similarly grouped data cluster. The training is done via minimizing the within-cluster sum of squares (WCSS) metric which uses squared Euclidean distances between the assigned centroid locations and observations. The centroid locations are then updated by calculating the new centroid location based on the observations assigned to the initial centroid assumption.

While k-means is easy to implement and its training is straightforward, it suffers from scalability issues where higher dimensional observations have poor clustering accuracy compared to the other methods. However, the advance of deep feature learning such as the autoencoder and convolutional autoencoder allowed for rich statistical representation of the input space at the code layers of deep neural networks which can be used with a k-means backend negating its drawbacks with high dimensional data. In this study the latent representation of the input images at the code layer of both the autoencoder and convolutional autoencoder structures are used as inputs to the k-means algorithm.

3.3 Experimental Setups

3.3.1 Datasets

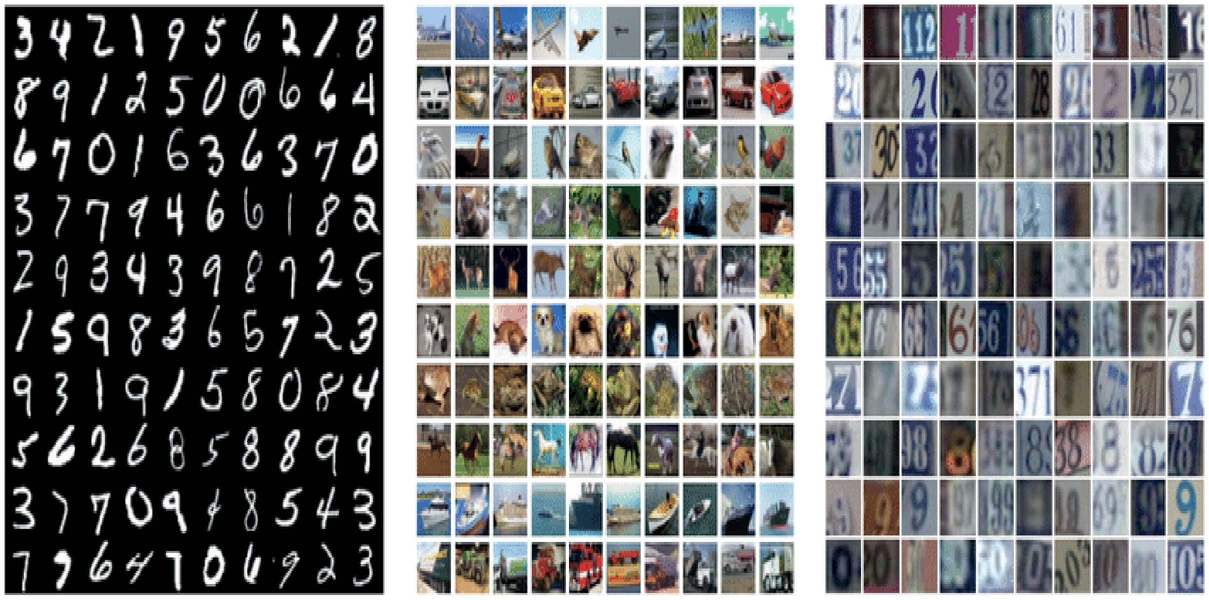


Figure 3.3: MNIST(a), Cifar10(b), and SVHN(c) Images Dataset Representation

We implemented the AE and CAE based neural network structures on three different datasets; MNIST, CIFAR10, and SVHN to analyze the effect of hyper-parameters on clustering performances and associated reconstruction losses.

- MNIST [59]: MNIST is a set of black and white handwritten image examples between 0 and 9 used as a popular benchmark dataset for deep learning applications in image classification. It has 60,000 images for training and 10,000 for testing where each image contains 28x28 pixels. Figure 3.3-(a) shows a collection of representative examples from this dataset.

- CIFAR10 [54]: Another popular benchmark dataset, CIFAR10, includes 10 classes with 6000 image samples per class to constitute 60,000 colored images of size 32x32 pixels where 50,000 images are used for training and 10,000 images are used for testing. The 10 different classes that are represented in the dataset include airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Figure 3.3-(b) shows a collection of representative examples from this dataset.

- SVHN [67]: Google's Street View House Numbers dataset consists of real-world images and includes 73,257 digits for training, 26,032 digits for testing with an image size of 32x32 pixels. The SVHN is similar to MNIST as it has 10 classes with numbers ranging from 0 to 9 except the images have color. Figure 3.3-(c) shows a collection of representative examples from this dataset.

3.3.2 Hyperparameters

The main hyper-parameters used in this study are the network topologies (both in terms of network size and input layer structure), size of the code layer (latent space) and clustering coefficients. In AE, the general network topology is 784-128-64-32-16. Four different code sizes are defined as 128, 64, 32, and 16 with five different clustering coefficients $K = 10, ..., 50$ for every 10 incremental of K. Unlike AE, we used a different network topology for different code sizes for the CAE as shown in Figure 3.4 with the same five clustering coefficients.

3.3.3 Evaluation Metrics

There are different clustering performance (evaluation) metrics defined in the literature separated as internal and external metrics such the Davies–Bouldin index [22] and Dunn index [28] for internal metrics, and Purity [86], Rand Index [79], F-measure [87], Jaccard Index [80], Dice Index [24], Fowlkes-Mallows Index [33], and Confusion Matrix [95] for external metrics for a variety of applications. We use the purity evaluation metric in this study to find the clustering accuracy for both algorithms for a fair comparison of the effect of hyper-parameters. To calculate the purity metric, each cluster is labeled as the group with the most frequent samples in that cluster, and the accuracy of this assignment is measured by

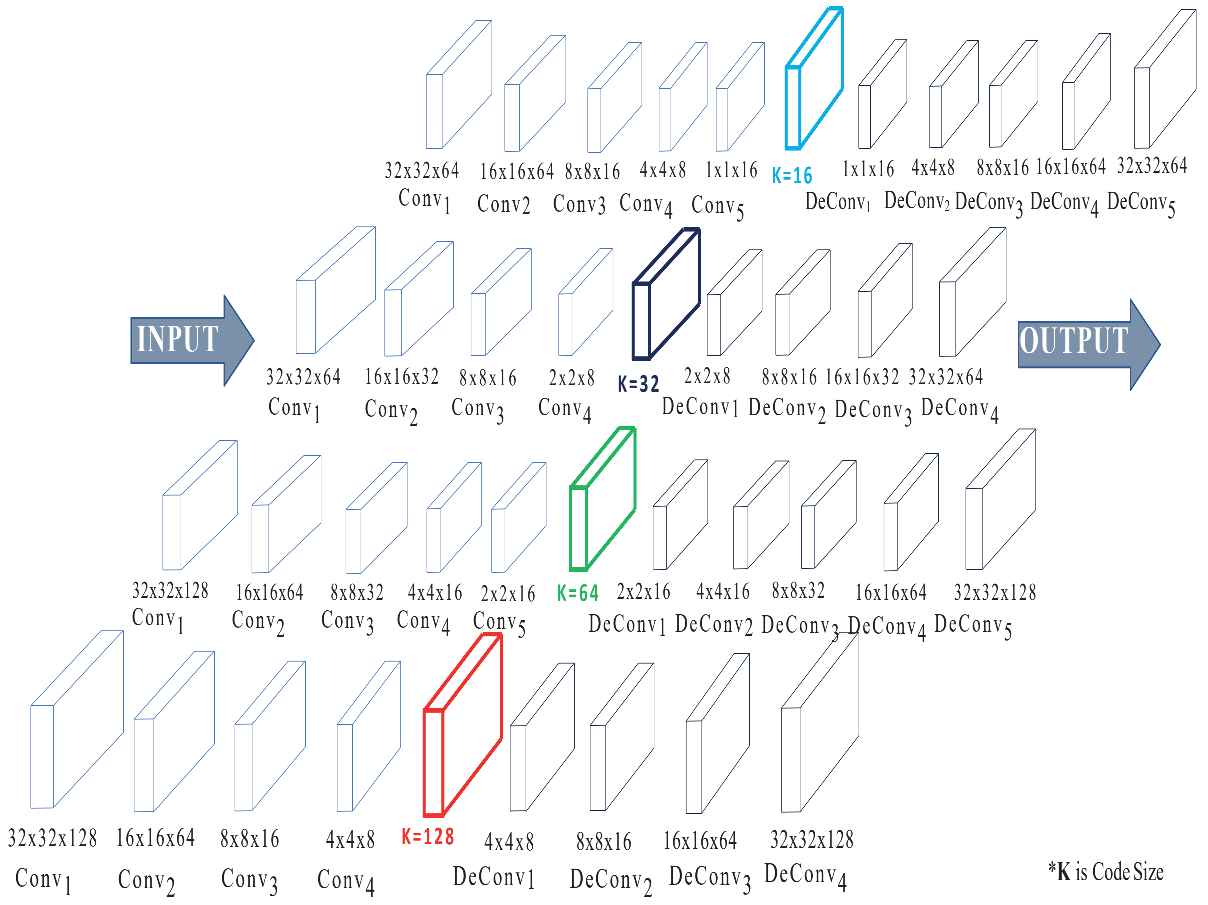


Figure 3.4: Implementation Representation of Convolutional Autoencoder Structures

finding the ratio of correctly assigned observations to the general population in that cluster for each group. Its formal definition is,

$$Purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{k} max_j |\omega \cap c_j| \tag{3.7}$$

where $\Omega = \omega_1, \omega_2, ..., \omega_k$ is the set of clusters and $\mathbb{C} = c_1, c_2, ..., c_j$ is the set of groups, N is the total number of data points.

3.3.4 Implementation

All implementations are done using the Keras [18] library on Google's Colab platform. There are differences in how the input data is represented based on the specific network topology being used. In the case of the autoencoder, the fully connected network topology for the encoder is D-128-64-32-16 where D is the input space dimension (feature space) size of 784 for the MNIST and 3072 for the CIFAR10/SVHN datasets and 16 is the code size (latent space). For a fair comparison of the datasets, CIFAR10 and SVHN are converted to gray-scale like MNIST (1D) prior to training on the autoencoder. After training the autoencoder, the latent representations of each observation in the dataset are collected in a transformed dataset (such as 60,000 x 16 for the MNIST dataset when using a code size of 16) on which the k-means clustering algorithm is applied to find the associated purity metric for each clustering coefficient (i.e., K = 10 through 50). The centroids are initialized randomly 20 times and the purity metrics are averaged to find statistically meaningful results. A similar process is repeated for the convolutional autoencoder except the colored images in CIFAR10 and SVHN are represented via the three RGB channels available at the convolutional front layer of this topology. In order to represent the samples from the MNIST dataset, the same image is presented to each channel creating a pseudo 3D representation for a fair comparison of the network structures. Figure 3.4 shows the detailed overview of each topology to obtain the desired bottleneck size for each experiment. We repeat the same procedure of applying k-means using different clustering coefficients on the transformed datasets for each code size.

3.4 Experimental Results and Discussions

The clustering performances of different network topologies on different image datasets for both autoencoder and convolutional autoencoder are presented in Table 3.1 and Table 3.2 below respectively.

Table 3.1: Convolutional Autoencoder for MNIST(Replicated), Cifar10, and SVHN

Cluster Size (K)	**Cifar10 (CAE)**				**Mnist (Grey-Scaled-CAE)**				**SVHN(CAE)**			
	Latent Space(k)				**Latent Space(k)**				**Latent Space(k)**			
	k=16	**k=32**	**k=64**	**k=128**	**k=16**	**k=32**	**k=64**	**k=128**	**k=16**	**k=32**	**k=64**	**k=128**
	Accuracy				**Accuracy**				**Accuracy**			
	Standard Deviation				**Standard Deviation**				**Standard Deviation**			
K=10	22.97	24.90	26.28	25.27	52.10	51.40	45.69	42.54	20.31	20.44	20.43	20.06
+/-	1.57	1.15	0.92	0.67	3.47	4.26	3.81	2.69	0.65	0.59	0.53	0.23
K=20	26.43	27.70	29.13	28.42	63.74	64.51	62.31	57.50	21.31	21.47	21.66	20.34
+/-	1.80	0.74	0.73	0.51	3.94	4.05	2.17	2.63	0.72	0.68	0.66	0.37
K=30	27.54	28.66	30.01	29.46	71.24	72.81	71.63	67.22	21.60	21.87	22.36	21.22
+/-	1.29	1.17	0.73	0.51	3.41	1.95	2.45	2.02	0.58	0.67	0.83	0.42
K=40	28.77	29.98	31.39	30.47	73.44	75.56	73.94	71.94	22.23	22.64	23.03	21.31
+/-	0.88	0.78	0.79	0.73	2.98	1.91	1.98	2.73	0.54	0.66	0.76	0.44
K=50	29.16	30.46	31.92	31.49	76.07	77.18	75.97	74.36	22.39	22.95	23.83	21.69
+/-	1.19	0.87	0.70	0.74	2.87	2.21	2.92	1.40	0.49	1.02	0.81	0.37

In Table 3.1, the following observations can be made. The highest clustering accuracy values are obtained pretty consistently at two different code sizes for the three different datasets. Where the maximum accuracy is observed at the code size of K = 32 for the MNIST dataset, a higher code size of K = 64 is needed for both CIFAR10 and SVHN datasets to achieve the highest clustering accuracy. This is expected due to the inherent complexity of the images of these two datasets when compared to MNIST. A similar trend is observed for supervised classification applications where the performances reported in the literature for MNIST are significantly higher than the ones reported for CIFAR10 and SVHN. Assuming that the back-end K-means algorithm perform similarly between different datasets; a larger code size is better able to capture the latent statistics of the more sophisticated images in CIFAR10 and SVHN. Another observation is that the clustering accuracy also increases as the K factor increases for the back-end clustering algorithm. This is also expected due to the performance metric used in this study (purity) which dictates that as the number of clusters increases, the probability of samples falling into a wrong cluster decrease. For instance, at the limit, when K is equal to the number of observations, the clustering accuracy would be 100% which would have no practical meaning. A standard practice in comparing clustering accuracies is to choose the K value to be either the same or twice the number of classes in the dataset.

Table 3.2: Autoencoder for MNIST, Cifar10(Greyscale), and SVHN(Greyscale)

Cluster Size (K)	**Cifar10(AE-Grey-Scaled) Latent Space(k)**				**Mnist(AE) Latent Space(k)**				**SVHN(AE-Grey-Scaled) Latent Space(k)**			
	k=16	**k=32**	**k=64**	**k=128**	**k=16**	**k=32**	**k=64**	**k=128**	**k=16**	**k=32**	**k=64**	**k=128**
	Accuracy				**Accuracy**				**Accuracy**			
	Standard Deviation				**Standard Deviation**				**Standard Deviation**			
K=10	20.42	20.71	20.90	20.95	64.04	66.57	66.52	59.23	20.05	19.86	19.92	19.83
+/-	0.93	0.74	0.59	0.45	3.39	1.76	1.06	2.21	0.47	0.23	0.32	0.23
K=20	23.24	24.31	23.49	23.81	73.23	74.99	73.56	70.06	20.90	20.94	20.72	20.46
+/-	1.12	0.53	0.86	0.45	2.20	2.04	1.53	2.63	0.68	0.70	0.65	0.30
K=30	24.71	24.31	24.84	24.74	78.09	79.80	77.86	76.47	21.74	21.68	21.87	21.48
+/-	1.05	0.74	0.57	0.45	2.16	1.40	1.17	1.83	1.16	0.66	0.60	0.31
K=40	25.33	25.13	25.58	25.70	81.46	82.47	81.27	79.26	22.68	22.49	22.47	22.11
+/-	0.99	0.65	0.79	0.48	1.53	1.14	1.03	1.65	1.53	0.52	0.64	0.28
K=50	26.39	26.69	26.57	26.49	82.21	83.52	83.27	81.05	23.21	23.44	23.47	22.53
+/-	0.86	1.65	0.59	0.62	1.67	0.73	1.19	1.61	1.12	0.86	0.95	0.37

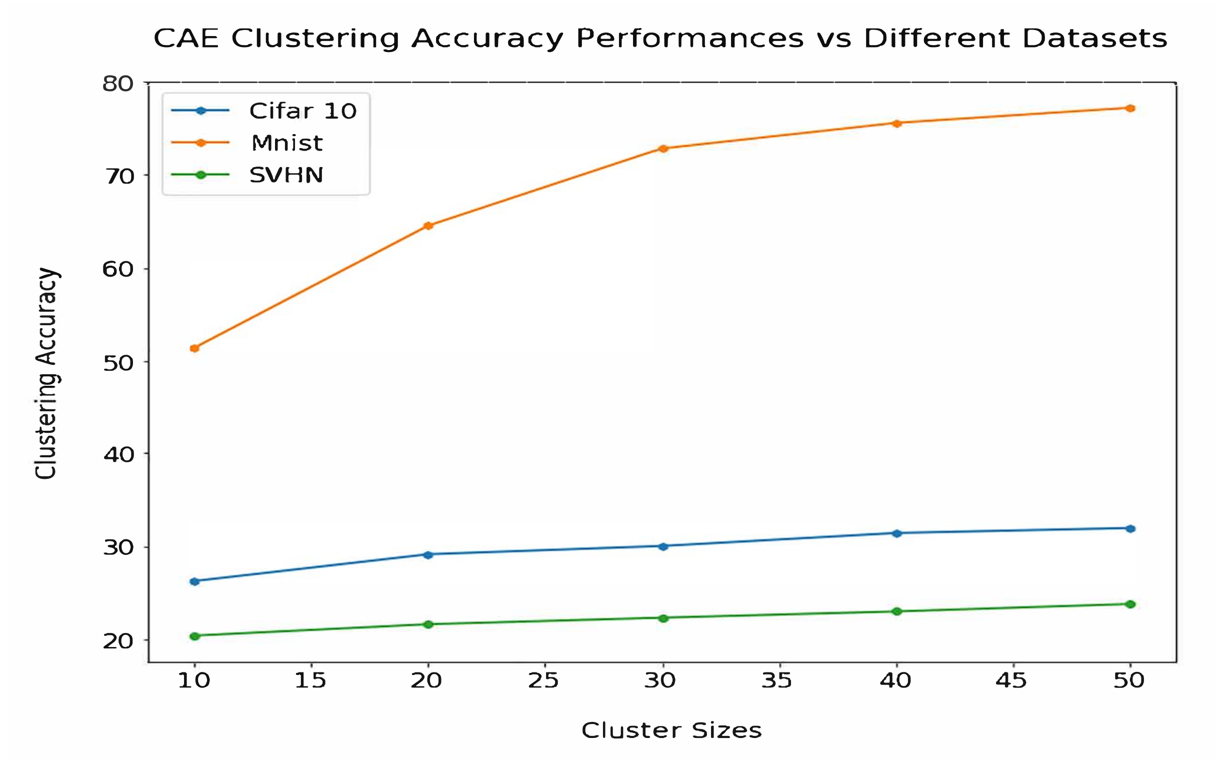


Figure 3.5: Convolutional Autoencoder Accuracies Across Different Settings and Datasets

In Table 3.2, a similar result is obtained for the MNIST dataset where the code size of k = 32 provides the highest clustering accuracy. In fact, the accuracy for the autoencoder in this case is greater than the accuracy reported for the convolutional autoencoder on the same dataset (as in Table 3.1). This can be explained by the fact that MNIST images have been replicated at the convolutional input layers designed for an RBG colored image which increases the number of parameters to be trained in the case of CAE which may have in turn reduced the maximum possible accuracy from the network topology due to a lesser ratio of observations to weight parameter comparatively. On the contrary, the clustering accuracies are lower for CIFAR10 and SVHN when using the regular autoencoder which suggests that the convolutional layers can properly utilize the additional information coming from the colored images. This effect is more significant for the CIFAR-10 dataset where color also signifies a meaningful feature of the observation (for instance a dog or a cat, as the two classes in the dataset, can only have a specific range of color values) compared to the SVHN

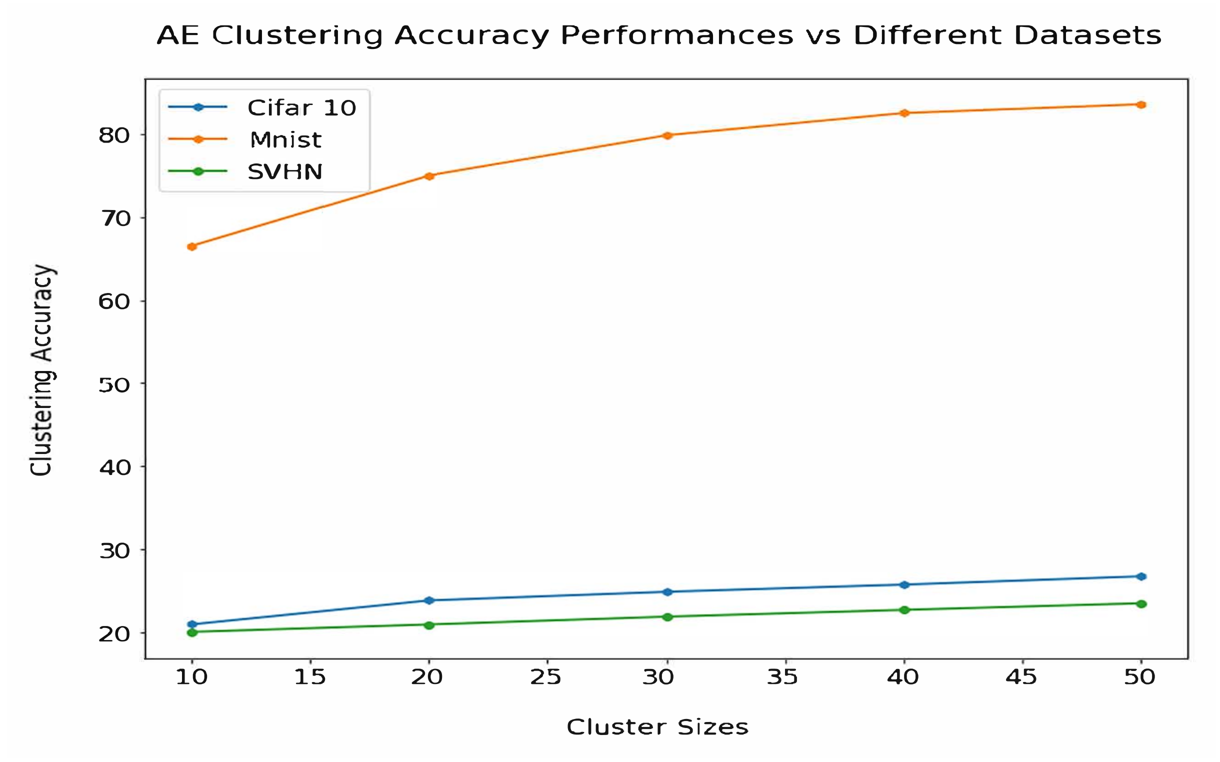


Figure 3.6: Autoencoder Accuracies Across Different Settings and Datasets

dataset where color is not as relevant. However, there is no standard code-size which provides the highest accuracy for all K values – which indicates that the lack of convolutional layers affects the code size required for maximum performance. In fact, for the CIFAR10 dataset, a code size of 128 (twice that of table 3.1) is generally required which may suggest that the information encoded in the convolutional layers is now represented (and compensated) in the increased bottleneck layer size. However, the results for SVHN do not support this conclusion where some of the high accuracies have been obtained at even lower code sizes such as 16. This is a very interesting observation which indicates that further research is required to understand such behavior and how removing convolutional layers could impact the training of the rest of the network for different datasets and how data is subsequently represented in the code layer. Figures 3.5 and 3.6 summarize the best performance curves for each topology as the latent space dimensions change.

Chapter 4: Anomaly Detection in Self-Organizing Networks: Conventional vs. Contemporary Machine Learning

4.1 Introduction

As a next-generation telecommunication technology, 5G brings a novel perspective and innovative solutions to the increased demand of humans and autonomous devices [13]. This technology mainly focuses on five areas, including dense-device structures, high carrier frequency bands such as millimeter wave (mmWave), multi-connectivity such as massive MIMO, smart devices, and massive machine-type communications. To fulfill these requirements in such a dynamic digital world, the next-generation cellular networks must be adaptable with predictive capabilities due to the ever changing environment of the nested services interacting with each other [5]. Hence, artificial intelligence (AI) has garnered increased interest as a potential 5G technology to handle the dynamic environment in analyzing and contributing to the execution of the network. There are examples of intelligent applications on massive machine-type communications (mMTC) in 5G or massive MIMO in wireless sensor networks (WSNs), to achieve better service quality through improved IoT connectivity as well as to extend battery life and boost spectral efficiency by utilizing channel aware decision fusion methodology [11],[20].

In previous mobile networks, such as 4G, autonomous mobile networks or self-organizing networks (SONs) with capabilities such as self-planning, self-configuring, self-optimizing, and self-healing, have been shown to significantly contribute to reducing network failures and boosting performance without human intervention such as AirHop's eSON [1]. However, current solutions in the market generally lack smart functionalities, especially for cell outage management (COM) to heal autonomically [23]. If there is no traffic due to a specific network issue, it signals an anomaly where one or more cells may be in outage and it is vital to detect the outage to resume service within the shortest time possible. For instance, Ericsson lost at least $100 million because of a network outage [30], which could have been prevented using AI-powered SONs. In consideration of these issues, the European Telecommunications Standards Institute (ETSI) introduced a zero-touch group to research automation to advance machine learning and AI techniques (deep learning) specifically for anomaly detection applications on mobile networks.

This paper has four main contributions to the field as summarized below:

- We present a comprehensive analysis of a conventional machine learning method for anomaly detection in self-organizing 5G networks (5G-SONs) and compare it with a popular deep learning alternative using different learning representations, including one-class and binary learning.

- We claim state-of-the-art performance on a publicly available dataset [9], which investigates multiple use case scenarios for anomaly detection in 5G-SONs. We demonstrate an average improvement of 15% over the best recent performance which was achieved by a deep auto-encoder-based setup.

- We demonstrate for the first time that data augmentation methods can further boost anomaly detection performance in binary mode, even when utilizing conventional algorithmic methods such as support vector machines on a sufficiently large dataset.

- Finally, we achieve nearly two orders of magnitude improvement in computational speed and an order of magnitude reduction in trainable parameters using conventional machine learning to provide a robust alternative for 5G self-organizing networks especially when the execution and detection times are critical.

The rest of this chapter is organized as follows. Section 4.1.1 provides a brief summary of prior work on anomaly detection in current mobile and self-organizing networks. Section

4.2 introduces the methods used in this study. Section 4.3 describes the experimental setup in detail and provides the hyper-parameters, dataset characteristics, implementation, evaluation metrics, and necessary details for repeatability. Finally, the results and discussions are presented in Section 4.4.

4.1.1 Related Research

Anomaly detection in communications has been an active research area over the last decade. For instance, in [76], abnormal activity in the wireless spectrum has been explored. Specifically, the authors used power spectral density (PSD) information to identify and pinpoint anomalies in the form of either undesired signals present in the licensed band or the absence of a desired signal. The information obtained from the PSD was processed using a combination of adversarial auto-encoders, convolutional neural networks, and long short-term memory recurrent neural networks.

In another example, [23] utilizes the measurements and handover statistics (inHO) from adjacent cells in a mobile communications network to expose abnormalities and outages. Monitoring in this way provides the potential status of a cell outage where the inHO information becomes zero. In [15], a novel online anomaly detection system was proposed in mobile networks to identify anomalies in key performance indicators (KPIs). The proposed system consists of a training and detection/tracking block. The system learns the most detrimental anomalies in the training block as each recent KPI is sourced, and monitors its status until the end of the second block. Thus, the detection of anomalies has been set to prefer highly possible anomalies in the long term. Moreover, the system objects to provide the minimum amount of anomalies by maintaining a low positive rate on behalf of network operators' efforts to deal with only real anomalies. In addition, the system can be extended to next-generation networks via automatic adaptation features to a new network behavior profile.

The authors in [65] proposed unsupervised learning to detect anomalies related to mobility using mobility robustness optimization (MRO), which is an important use case of SONs for modern 4G and 5G networks. A similar study in [75] brings a different perspective by using reinforcement learning to degrade call drop rates.

In [17], the authors used an autoencoder-based network anomaly detection method to detect the ratio of changes in features that reflect non-linearity to increase detection accuracy. The authors used a convolutional autoencoder for dimensionality reduction and outperformed more conventional methods in wireless communications to detect cyberattacks.

Artificial intelligence (AI) applications supporting 5G technologies can be implemented both in the physical and network layers. For instance, authors in [69] present an excellent overview of deep learning (DL) applications for the physical layer. One of the applications is a novel modification of the standard autoencoder called the "channel autoencoder". In a typical autoencoder, the goal is to find the most compressed representation of the inputs in the encoding layer with minimal loss at the output layer. In [69], the authors aim to find the most robust representations of the input (messages) to account for the channel degradation by adding redundancies instead of removing them. The authors further extend this concept to an adversarial network of multiple transmitter/receiver pairs aiming for increased capacity. Furthermore, they discuss augmented DL models using radio transformer networks (RTNs). The RTNs carry channel domain information and simplify the receiver's job for symbol detection using a neural network estimator to obtain the best parameters for symbol detection. The augmented DL models on complex I/Q samples for modulation classification demonstrated that the DL models outperformed the classification methods based on expert features. Another study discussed in [2] introduces designing mobile traffic classifiers based on the DL utilization. A systematic framework of new DL-adopted Traffic Classification (TC) structures is introduced and analyzed. Rather than mobility, the study includes a wide allocation range to encrypt the TC.

These studies generally rely on multi-class applications of deep neural network methods learning on features such as key performance indicators, handover statistics, reference signal received power and quality (RSRP and RSRQ), number of connection drops and failures. However, there exists a discussion in the research community where other learning representations such as one-class learning, and deep unsupervised methods have the potential to become strong alternatives for anomaly detection in SONs [9]. Similarly, an ever-present debate investigates the comparative effectiveness of empirical deep learning models and conventional approaches such as feature engineering for a range of temporal applications [51]. A categorical analysis of literature discussed in this section can be found in Table 4.1, which displays the important characteristics and features such as the dataset type, topology of the network used, which learning methodology is applied for which application, etc.

Table 4.1: Summary of Related Research

Paper Name and Reference	**Features**	**Dataset**	**Topology**	**Learning**	**Application**	**Data Preprocessing**
Asghar et. al. [9]	KPIs (RSRP, RSRQ)	User-Cell Based Four Use Case Datasets	AE (Autoencoder), k-NN (k-Nearest Neighbor)	Unsupervised	Cell Outage Detection	Yes
Rajendran et. al. [76]	PSD (Power Spectral Density)	One Synthetic Spectrum Dataset, Two Real Wireless Datasets (SDR, PSD Sensor Data)	VAE (Variational Autoencoder), AAE(Adversarial Autoencoder), LSTM(Long-Short-Term Memory)	Unsupervised/ Semi-Supervised	SAIFE (Spectrum Anomaly Detector with Interpretable Features)	Yes
Bandera et. al. [23]	InHO-KPIs (Incoming Handover Statistics – Key Performance Indicators)	Live LTE Network Simulator	LTE Cellular Layout	Unsupervised	COD (Cell Outage Detection)	No
Burgueño et. al. [15]	KPI (Key Performance Indicators)	Real LTE Advanced Mobile Network	DBSCAN (Density based spatial clustering)	Supervised	Online Anomaly Detection	Yes

Table 4.1 (Continued)

Moysen et. al. [65]	PM (Performance Management), CM (Configuration Management)/ IM (Inventory Management)	Commercial LTE Networks Datasets	PCA (Principal Component Analysis), HDBSCAN (Hierarchical Density Based Spatial Clustering of Applications with Noise)	Unsupervised	Identifying Cells Experiencing Performance Degradation	Yes
Chen et. al. [17]	TCP (Transmission Control Protocol), UDP(User Datagram Protocol), ICMP (Internet Control Message Protocol) Traffics	NSL-KDD	PCA (Principal Component Analysis), AE (Autoencoder), CAE (Convolutional Autoencoder), k-NN (k-Nearest Neighbor), SVM (Support Vector Machine), TANN (Triangle Area Based Nearest Neighbors)	Unsupervised/ Supervised	Network Anomaly Detection	Yes

Table 4.1 (Continued)

Kanjilal et. al. [51]	ADL (Activities of Daily Living)	UniMIB-SHAR	SVM (Support Vector Machine), ANN (Artificial Neural Network), LSTM (Long-Short-Term CNN (Convolutional Neural Network)	Unsupervised	Human Activity Recognition	Yes
O”Shea et. al. [69]	Real Signal Messages, I/Q Samples	RML2016.10b	AE (Autoencoder), RTN (Radio Transformer Networks), CNN (Convolutional Neural Network)	Unsupervised	Deep Learning Applications on Physical Layer-Survey	Yes
Aceto et. al [2]	HTTP Traffic, User-Website Fingerprint, Flow-based	Multi-Class Dataset, FB/FBM (Face-book/ Facebook Messenger) Binary Dataset	SAE (Sparse Autoencoder, CNN, LSTM (Long-Short Term Memory)	Unsupervised/ Supervised	Real Human Users' Activity	Yes

4.2 Methods

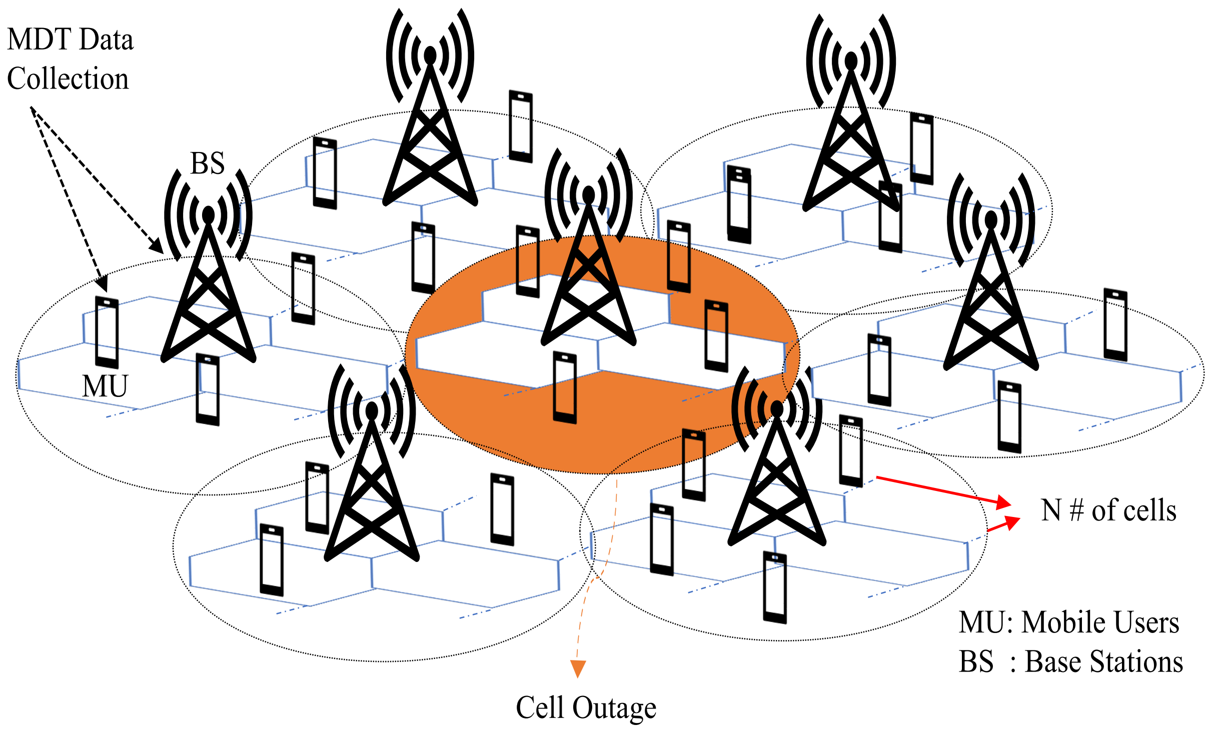


Figure 4.1: Grid Based Layout Used in the Simulation of Cell Outage Detection

4.2.1 Anomaly Detection Problem Definitions

In this paper, we look at anomaly detection scenarios where a total of 105 mobile users are uniformly spread around 7 base stations (BS) and the BS can fail to communicate with the user by performing below the expected key performance indicator (KPI) margins. Each BS is represented as a circle and in each circle, there are 3 cells regularly spread out shown in Figure 4.1, for a total of 21 cells. Minimization of drive test (MDT) report is generated for each user at a 5kHz sampling rate (i.e., once every 0.2 ms) with a total recording time of 50 seconds to obtain the associated data. The KPIs including the RSRP and RSRQ measurements are collected from each of the 7 base stations at this time. The data for each MDT report is then assigned a class label from the numerical set 0,1,2,3.

The three labels; 0,1, and 2 indicate normal network status with varying levels of discrete transmitter power, whereas label 3 indicates that the base transceiver station (BTS) has failed to communicate with the user, which is classified as an anomaly in the dataset. Three other scenarios are implemented in a similar way with the only differences being the numbers of users (210, 105, 105) and cells (57, 57, 57) to be able to consider the anomaly detection (AD) application for a wider range of networks and users with different numbers of observations.

Figure 4.1 shows the structure of the communications network for the case of 7 BS and 3 cells within each grid used in the first scenario for outage detection. This figure is provided to help with the visualization of the AD application and is similar for different scenarios using a range of BS, cell and user numbers.

4.2.2 Support Vector Machine

The support vector machine (SVM) [91] is a popular machine learning algorithm that plays important roles in binary classification and unsupervised feature learning. An SVM model represents and separates various classes by building a set of hyperplanes in a multi-dimensional space. Separation of the classes from each other is performed iteratively by the SVM algorithm through finding the optimal hyperplane (the decision region between a group of objects from different classes). In this context, the optimal hyperplane maximizes the distance gap (margin) between the two lines closest to the data points from different classes based on the support vectors (data sample points where the classification error is minimum).

We implement SVM in two different ways:

4.2.2.1 One Class Anomaly Detection with SVM

One-class SVM is a popular method for unsupervised anomaly detection [97, 21, 14, 72]. In general, an SVM is applied to binary classification tasks. In the one-class anomaly detection case, the algorithm is trained only with observations from the majority class to learn the "normal" samples . When new data are presented to the SVM algorithm the "anomaly" samples generate a lower decision probability compared to the observations from the majority class under ideal conditions.

4.2.2.2 Binary Anomaly Detection with SVM

A binary SVM classifier provides the most accurate results when trained on balanced datasets. When the dataset is imbalanced, the SVM classifier model is inclined to overfit to the majority class and shows poor performance for the minority class with reduced generalization. To overcome the imbalance in a dataset, different techniques have been introduced in the past one of which, the synthetic minority oversampling technique or SMOTE, has gained increased popularity in cases of severe imbalances such as anomaly detection.

4.2.3 Smote Algorithm

The synthetic minority oversampling technique, or SMOTE [16], is one of the most popular methods for oversampling to overcome the imbalance problem. Its goal is to balance class distribution by randomly increasing the minority class by generating 'synthetic' patterns based on features instead of raw data. The oversampling process of the minority class (**C**) begins by selecting each minority class sample (**X**), where $\mathbf{C} \epsilon \mathbf{X}$ and interpolating synthetic instances along the lines that connect minority instances and their k-nearest neighbors **x**. The k value is chosen randomly according to the hyper-parameter oversampling rate **N** based on the number of minority samples. First, a distance-based method, such as the 'Euclidean distance', is used to calculate the distance between the feature vector and its neighbors. Second, the distance is multiplied by a random number between (0,1] and added to the previous feature sample.

Hence, new and synthetic features were generated along the line segments between the two original samples. Mathematically:

$$X' = X + rand(0,1)^{*}|X - X_k| \tag{4.1}$$

where X' is the new set of synthetic samples, and X_k is the set of randomly selected k-nearest neighbor samples.

4.2.4 Autoencoder

Autoencoder is a particular artificial neural network topology which has the same input and output layers where the training is performed by presenting the same input data to both layers simultaneously. The general structure of the auto-encoder consists of a visible input layer $\mathbf{x}$, a number of hidden layers $\mathbf{h}$ and the reconstructed output layer $\mathbf{y}$ with a family of nonlinear activation functions $\mathbf{f}$ applied at different layers.

During training, the auto-encoder maps the input $\mathbf{x} \varepsilon \mathbf{R^y}$ to the hidden layers with lesser dimensions than the input data which produces a compressed representation of the original data in which its dimensionality is reduced to the code(latent) layer size $\mathbf{H} \varepsilon \mathbf{R^h}$. This first step is called the "encoder" which provides the compressed information to be mapped to the output layer via the "decoder," through a process called "reconstruction." Mathematically, these two steps are formulated as follows:

$$H \equiv f_{WH}(x) = f(W_H x + b_H) \tag{4.2}$$

$$z \equiv g_{Wz}(x) = g(W_z H + b_Z) \tag{4.3}$$

where W_H and W_Z define the encoding weight and decoding weight, respectively, b_H and b_Z define the corresponding encoding bias vector and the decoding bias vector, and $f(.)$ and $g(.)$ are encoding and decoding activation functions respectively such as a sigmoid function or a rectified linear unit. As previously mentioned, the primary purpose of the autoencoder is to learn useful latent information on the coding layer by minimizing the reconstruction error. For a given N input data samples, the following loss function is used to determine the parameters "W_H, W_Z, b_H, and b_Z" through a back-propagation algorithm commonly used in feed-forward neural networks:

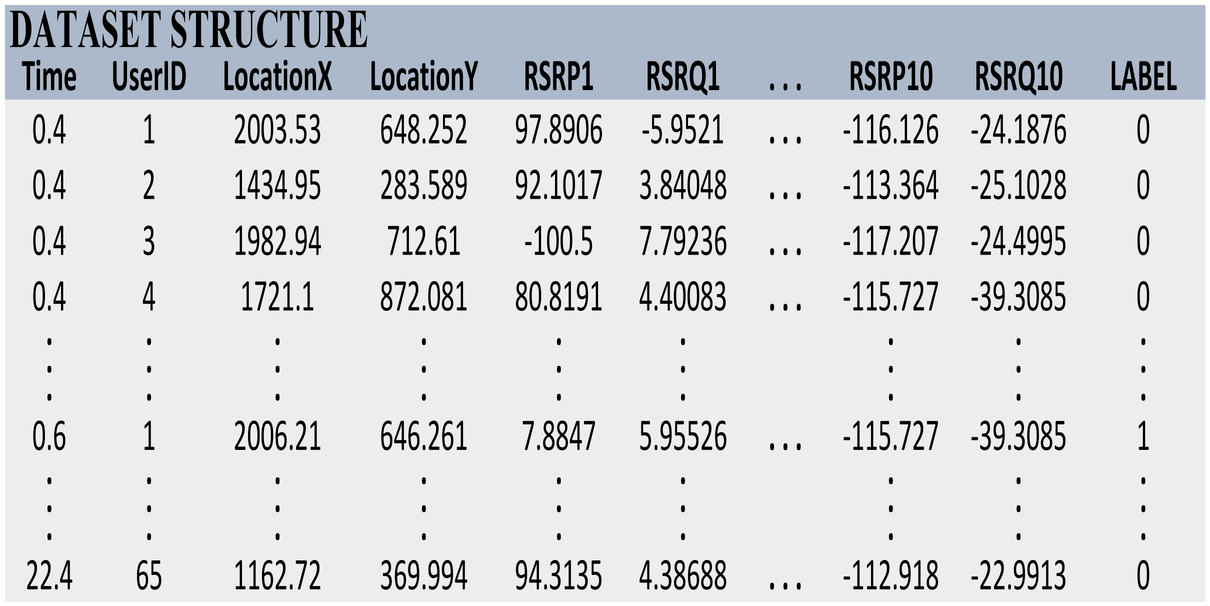
$$L_{AE} = min\frac{1}{N}\sum_{k=1}^{N}||x_k - z_k||^2 \tag{4.4}$$

Within the context of this application, the autoencoder is used for anomaly detection by training the network using normal observations and once trained, comparing the reconstruction error of the normal and anomalous samples to a threshold for detection. It is hypothesized that the normal samples will provide lesser reconstruction errors compared to the anomalous samples which can easily be characterized using the receiever-operating-characteristics (ROC). More specifically, in [9], the autoencoder model structure start with an input vector of size 20 (which corresponds to 10 RSRP and 10 RSRQ measurements), followed by four hidden layers including 12, 6, 6 and 12 neurons to implement a general topology as follows: 20-12-6-6-12-20.

4.3 Experimental Setups

4.3.1 Datasets

In this work, we use the dataset created by a SON simulator previously introduced in [9] and made available to the public for further research [8]. The dataset includes four different application scenarios where the data is collected periodically (with a 5 kHz sampling rate) from a minimization of drive test report [48], which includes mobile user information regarding the user activities recorded in the enclosed regions around the base stations (cells) in certain measurement periods. There are four different datasets with different numbers of users and use-cases. Figure 4.2 demonstrates the basic structure of the dataset for the first use case (dataset 1), which consists of the measurement time, a unique ID assigned to each



DATASET STRUCTURE

Time	UserID	LocationX	LocationY	RSRP1	RSRQ1	...	RSRP10	RSRQ10	LABEL
0.4	1	2003.53	648.252	97.8906	-5.9521	...	-116.126	-24.1876	0
0.4	2	1434.95	283.589	92.1017	3.84048	...	-113.364	-25.1028	0
0.4	3	1982.94	712.61	-100.5	7.79236	...	-117.207	-24.4995	0
0.4	4	1721.1	872.081	80.8191	4.40083	...	-115.727	-39.3085	0
⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮
0.6	1	2006.21	646.261	7.8847	5.95526	...	-115.727	-39.3085	1
⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮
22.4	65	1162.72	369.994	94.3135	4.38688	...	-112.918	-22.9913	0

Figure 4.2: Dataset Structure for the First Use Case

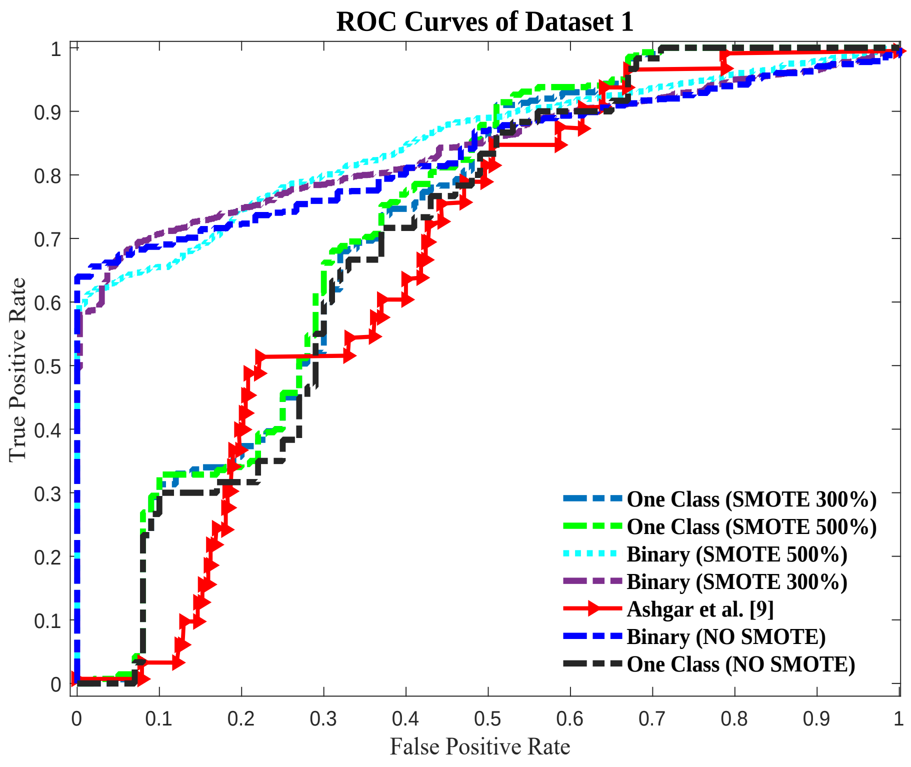


Figure 4.3: Dataset 1 AUC Curves

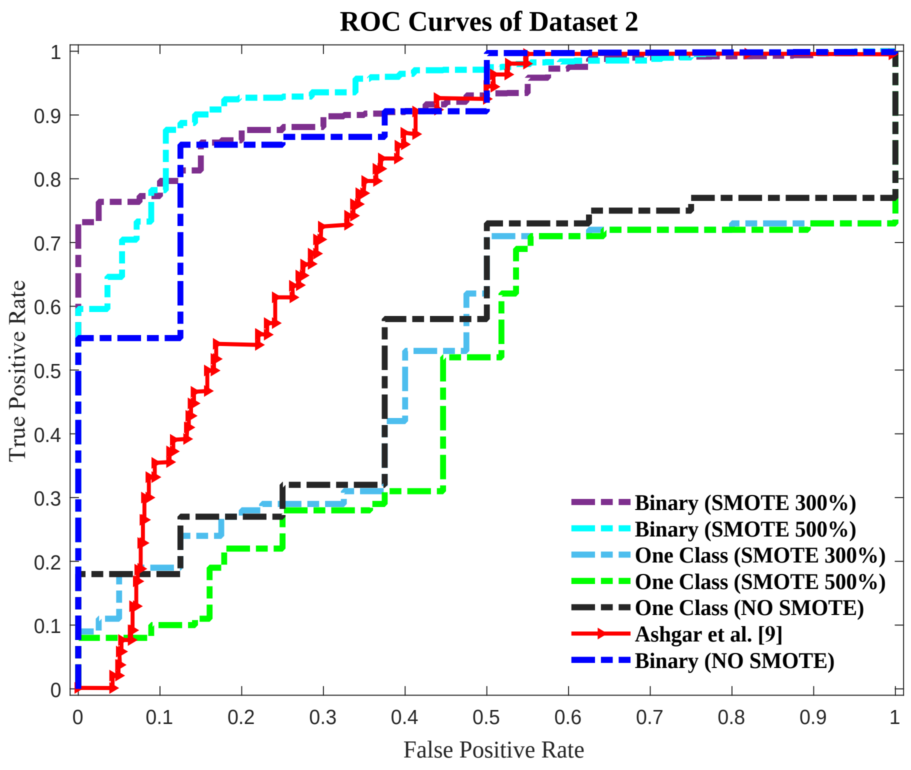


Figure 4.4: Dataset 2 AUC Curves

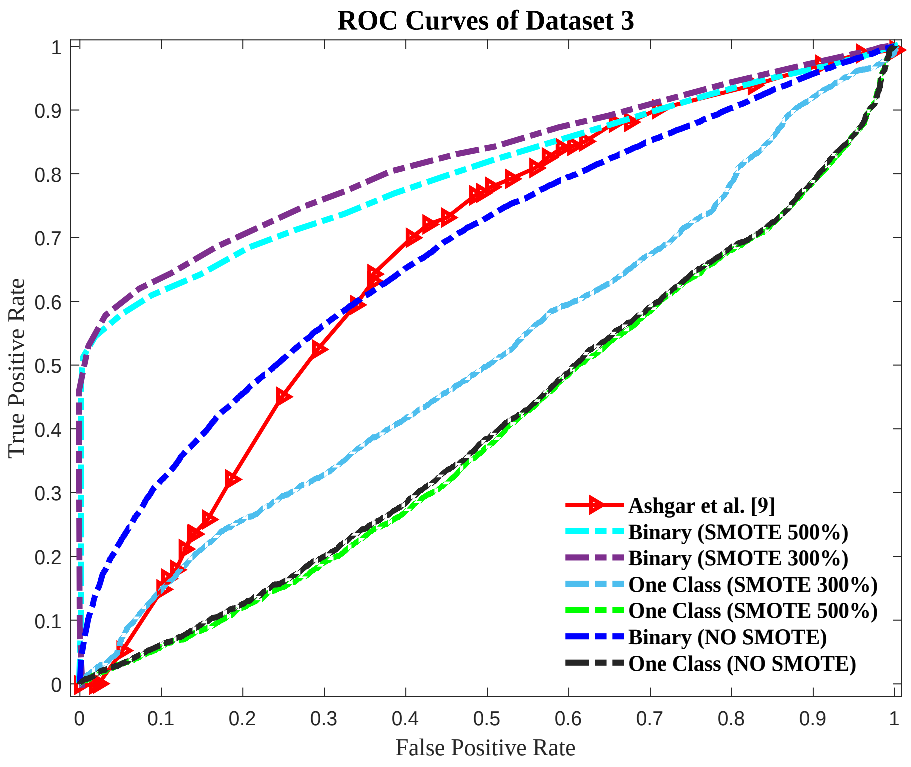


Figure 4.5: Dataset 3 AUC Curves

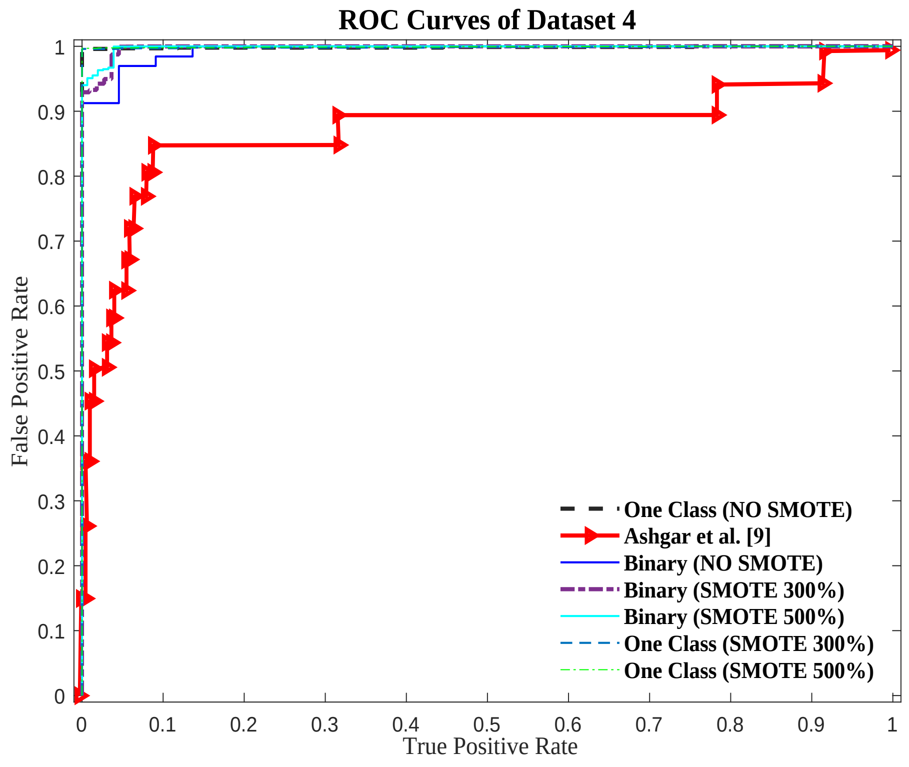


Figure 4.6: Dataset 4 AUC Curves

user, the coordinates of user locations in two dimensions at the time of measurement, the reference signal received power (RSRP), the reference signal received quality (RSRQ), and the label that shows if the associated entry (i.e., the collection of measurements associated with that user) is anomalous (1) or not (0). RSRP and RSRQ are significant measures of signal level and quality in LTE networks designated as key performance indicators (KPIs) in identifying whether the collected information is anomalous. The feature vector consists of the RSRP and RSRQ measurements from different cell locations (i.e., RSRP1, RSRQ1, RSRP2, RSRQ2, . . . , RSRP10, RSRQ10). The dataset has 11674 observations where only 60 of them are anomalous (i.e., 1 in 200) measurement samples and 25 features including the feature vector, user ID, location, and class labels.

The second dataset is similar to the first one in terms of the number of features and measurements except that it has a much less frequent anomaly rate with 8382 observations, where only eight of them are anomalous (i.e., approximately 1 in 1000).

Datasets 3 and 4 represent different use cases, both of which have 114 features with a longer (80s) recording time, which resulted in a much larger observation base (42000). As in datasets 1 and 2, they represent different anomaly rates where dataset 3 has a much larger sampling of anomalous measurements (9635) compared to dataset 4 (22) because only the entries below the -120 dB RSRP measurement threshold were labeled as such.

4.3.2 Hyper-parameters

The most significant hyper-parameters in this study are the oversampling rate N and the number of nearest neighbors k, specifically for the SMOTE algorithm in applying oversampling for the binary classification case. In applying SMOTE, we tested two different sets of hyper-parameters with N = 300, k = 4 and N = 500, k = 5.

4.3.3 Implementation

All implementations were performed using MATLAB 2020b. We tested both one-class and binary SVM models on all datasets. The SMOTE algorithm was used with the binary SVM model to adjust for the imbalanced datasets. The datasets were preprocessed, where the time, UserID, and location features were not included in the training process for fairness. Approximately 10 % of the normal and anomalous samples were separated for testing. Both the training and testing samples were normalized to between (0,1].

The one-class SVM model was trained with only normal samples using Gaussian RBF kernels. After training, we generated the SVM probability outputs with test samples, including both normal and anomalous samples, to obtain receiver operating characteristic (ROC) curves along with area-under-the-curve (AUC) scores as performance metrics.

The binary SVM model is trained in exactly the same fashion except that in addition to the above process, the anomaly samples are oversampled with SMOTE to generate balanced datasets prior to training and testing the algorithm.

4.3.4 Evaluation Metrics

In this study, we evaluated performance by looking at ROC curves and AUC scores. The ROC is a probability curve that shows the model's ability to identify the positive class appropriately. It is plotted with the true positive rate (TPR) on the y-axis and false positive rate (FPR) on the x-axis, where TPR is the percentage of correctly classified positive outputs and FPR is the percentage of incorrectly classified positive outputs, as expressed below:

$$TPR = \frac{TP}{TP + FP} \tag{4.5}$$

$$FPR = \frac{FP}{FP + TN} \tag{4.6}$$

The AUC, on the other hand, provides a summarized number as an indication of how powerful the model is in discriminating between classes with the mathematical expression as below:

$$AUC = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.7}$$

Table 4.2: Representation of AUC Scores of All Datasets for One-Class SVM and Binary SVM Classifiers

			Dataset 1	Dataset 2	Dataset 3	Dataset 4
SVM	Binary (No Smote)		84%	90%	68%	99%
	One Class (No Smote)		69.52	58.125%	40.92%	99.84%
Smote Algorithm	N=500% K=5	One Class	72.52%	47.30%	30.40%	99.91%
		Binary	**87%**	**94%**	**82**	99%
	N=300% K=4	One Class	71.76%	49.72%	51.20%	99.86%
		Binary	83%	93%	80%	**100%**
Ashgar et al. AUC [6]			67.25%	78.49%	66.54%	87%

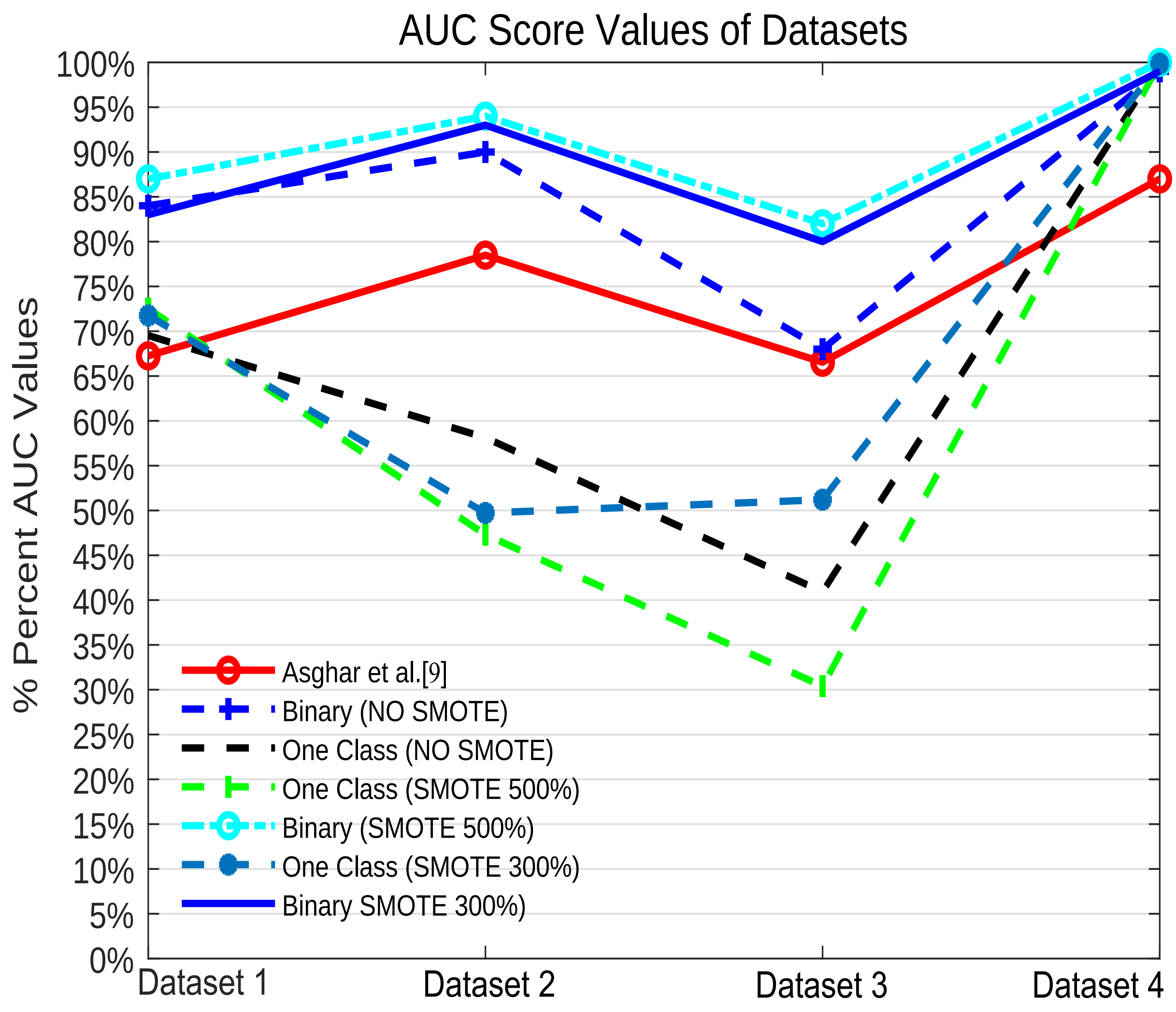


Figure 4.7: Area Under Curve (AUC) Scores Across Different Datasets

4.4 Results and Discussion

The ROC curves for each of the four datasets are shown in Figures 4.3 through 4.6. The red curve in each figure represents the latest state-of-the-art reported on this dataset using a deep autoencoder [9]. For comparison, different SVM implementations are represented in different colors, including binary and one-class combinations with and without augmentation using SMOTE. The results are generally consistent except in the case of dataset 4, where all SVM combinations outperformed the deep autoencoder at all levels of TPR & FPR. In the case of the first three datasets, the deep autoencoder outperformed the SVM implementations without augmentation. However, when SMOTE is used, both one-class and binary modalities of the SVM demonstrate higher performance compared to the original study, and in some cases significantly so.

Table 4.3: Abbreviations Used in This Study

Abbreviation	**Meaning**
AI	Artificial Intelligence
AUC	Area Under the Curve
BTS	Base Transceiver Station
COM	Cell-Outage Management
DL	Deep Learning
ETSI	European Telecommunications Standards Institute
FB/FBM	Facebook/Facebook Messenger
FPR	False Positive Rate
inHO	Measurements and Handover Statistics
IoT	Internet of Things
5G-SON	Self-Organizing 5G Networks
KPIs	Key Performance Indicators
LTE	Long-Term Evaluation
MDT	Minimization of Drive Test
MIMO	Multiple Input Multiple Output
Mm Wave	Millimeter Wave
MRO	Mobility Robustness Optimization
PSD	Power Spectral Density
RBF	Radial Basis Function
ROC	Receiver Operating Characteristics
RSRQ	Reference Signal Received Quality
RSRP	Reference Signal Received Power
RTNs	Radio Transformers Networks
SMOTE	Synthetic Minority Oversampling Technique
SONs	Self-Organizing Networks
SVM	Support Vector Machine
TC	Traffic Classifier
TPR	True Positive Rate

The results are further summarized in Figure 4.7, which provides an overview of AUC scores across different datasets and algorithms where both binary and one-class SVMs (with the three curves at the top) clearly outperform the deep autoencoder approach shown in red. Table 4.2 provides the absolute numbers in terms of the AUC with 5% significance. The best combination of SMOTE and SVM modality outperformed the deep autoencoder by 19.75%, 15.5%, 15.5%, and 13% for datasets 1, 2, 3, and 4, respectively. This corresponds to an average performance improvement of over 15% across all the application scenarios. It is important to note that even without artificial augmentation of the dataset, conventional machine learning using SVM still outperforms the previous state-of-the-art methods reported in the literature. However, the difference in performance was noticeably less.

4.4.1 Computational Complexity Analysis

Computational complexity analysis is an important step in identifying the strengths and weaknesses of conventional algorithm such as one-class and binary SVMs when compared to the more modern approaches such as the autoencoders used in anomaly detection. In this paper we focused on both the raw computational times specifically for testing phase for each of the four datasets scenarios as well as the number of trainable parameters for both algorithms. We also compared how SMOTE affected the complexity.

All measurements are done using the latest version of MATLAB at the time of this writing (2021b) using the standard time measurement scripts. On the first dataset, for the one-class SVM algorithm, the testing time took 90 ms without SMOTE and 140 ms with SMOTE, whereas it took 9000 ms to run the autoencoder, almost a 100-fold increase. Additional complexity of SMOTE is more than outclassed by its significant contribution to the accuracy. We have observed similar computational times for the rest of the use-case scenarios (i.e., for the second dataset one-class SVM testing times were 110 - 130 ms (SMOTE) compared to 7120 ms for the AE) where there were orders of magnitude improvement in testing times. We have performed the computational speed analysis using one-class SVM due to the fact that the implementation was done using a native script whereas for the binary SVM, a GUI toolbox was used with better visualization capabilities which affects speed. However, there should be no difference in testing times since the SVM topologies are identical with the only difference being the way the data is represented to the algorithms.

In terms of the trainable parameters as an indicator of computational complexity, the SVM models on average had 23 support vectors compared to the 660 weight and 76 bias parameters (total of 736) that need to be trained for the autoencoder. Based on the above measurements for computational times and complexity, SVM-based approaches are less complex even when using SMOTE, have competitive AUC scores, and are thus more suitable for time-critical scenarios such as anomaly detection / outage recovery compared to the AE.

Future work will study the impact of augmentation on other learning algorithms, specifically statistical deep learning, such as variational auto-encoders. Work presented in this paper can further be extended to other applications beyond anomaly or outage detection. Specifically, there has been increased attention to modulation detection in next generation mobile wireless networks where fast, robust, and light machine learning models could enable time-critical applications in signal classification and modulation detection. Improvements in speed can be realized both at the algorithm level and data preprocessing stages using techniques such as principal component analysis to identify the most relevant features for classification and detection. Finally, statistical learning algorithms, such as Gaussian Process Regression, which have gained immense popularity as alternatives to deep learning can be applied to different scenarios especially when data is not present in sufficiently large volumes to properly train DL models with many parameters.

Chapter 5: Fast, Robust, and Light Machine Learning for Signal Classification in Next Generation Mobile Networks

5.1 Introduction

Today's biggest challenges for wireless communications systems include growing demands on data transmission capacity, integration of multiple families of devices with increasing population and network optimization. Therefore, supplying the increased demand within a limited spectrum for rapidly changing environments in the next-generation wireless networks makes the cognition of intelligent systems more critical than ever. Both the FCC and the researchers in academia and industry continue to emphasize the importance of advanced spectrum sensing and signal identification to improve the quality of service while considering integrated systems and services. Within this framework, the next-generation communication devices with intelligent features are being prospected to recognize and detect over-the-air signals and choose the most suitable transmission layout.

Signal identification methods are studied from two main perspectives: likelihood-based [27] and feature-based [26]. Although the likelihood-based methods optimize the decision threshold, they suffer from high complexities and model mismatch. On the other hand, the feature-based methods extract unique attributes from the received signal, such as higher-order cyclic-related statistics, where some of these features may be are more sensitive to the environmental conditions.

In cognitive radio environments, over-the-air signal identification [70] has been employed in various fields including the military (i.e., directing energy to control spectrum) and civilian (i.e., RF spectrum monitoring) communication systems. Among the aforementioned methods, the ones with cyclostationary-based features carry the most robust characteristics for real-world conditions, making them the best candidates for modulation classification. For instance, modulations such as phase-shift keying (PSK), frequency-shift keying (FSK), and quadrature-amplitude modulation (QAM) can be separated using cyclostationary attributes [35]. For instance, cyclostationary feature detection in [78] can distinguish generic modulation techniques. Furthermore, second-order cyclostationarity [52] can be used for the classification of LTE and GSM signals in radio access technology identification. A tree-based classification method [29] is proposed for 3GPP-designed signal classification (i.e., GSM, CDMA2000, UMTS, LTE). Such classical methods depend on specific features for employing the likelihood-based techniques in the statistical decision process. As a result, the decision parameters, including the threshold and number of samples need to be adjusted for adaptive situations in a real-life application. Moreover, to satisfy the standards for the next-generation wireless networks [44] one has to tolerate and take into account such deficiencies of the classical methods in real-world conditions. In response, recent data-focused AI-based approaches [92] were shown to address some of the most significant shortcomings for classical modulation detection in wireless communications.

Machine learning and deep learning (when there is sufficient data to train a model properly) have been used to reduce the variability of the environmental conditions in signal identification. More specifically, data-centric approaches such as feed-forward neural networks (FNN), support vector machines (SVM) and convolutional neural networks (CNN) have been demonstrated to have success in this application. In [74], a multilayer perceptron is employed to identify twelve modulation types with high accuracy over a broad range of signal-to-noise ratio (SNR) values. In [83], two distinct FNN topologies are implemented using standard gradient descent to classify eight modulation types with high accuracy at low SNR conditions. In [31], AM and PSK modulations are identified by employing a neural network architecture fed with spectral correlation function (SCF)-based features. In [62], an SVM is utilized to estimate the optical signal-to-noise-ratio (OSNR) and modulation types by using cumulative distribution function (CDF) -based features to reduce the dimensionality of the feature space. Similarly, in [61], principal component analysis (PCA) is used to reduce the feature size using a wavelet transform. In [93], a CNN classifier is trained on SCF-based features as a deep learning application.

Most studies focused on improving classification accuracies for signal identification. While most claim deep learning is the state-of-the-art in multi-class modulation identification application such as [93], it is generally more complex than conventional machine learning and the identification process takes longer where timing is of critical importance [77]. In this research we investigate the promise of conventional data-centric approaches to signal identification at different SNR values and compare with the latest state-of-the-art on a real-world dataset. We focus on reducing the identification time by both reducing the number of trainable parameters in the model and applying data compression techniques such as PCA to reduce the vast temporal dimension of the feature set. The classifier models are capable of classifying AWGN, UMTS, GSM, LTE signal types by using the dataset given in the SCF format as the future matrix. We demonstrate that the proposed network with or without using data compression outperforms current state-of-the-art, recently reported in *the IEEE Transactions of Vehicular Technology* in correctly identifying UMTS, GSM, and LTE modulations including AWGN as background using SCF based features with significantly faster training and identification [93].

5.2 Analytical Framework

The ultimate objective of the proposed algorithm is to identify the modulation of the complex baseband signal x(t) in a fading channel where the received signal r(t) with AWGN w(t) is defined as follows:

$$r(t) = l(t) * x(t) + \omega(t), \tag{5.1}$$

where $l(t)$ is the impulse response of the time-varying channel and * shows the convolution process [92]. In this study, x(t) is identified through a data-centric approach using the cyclostationary signal processing on r(t) within the multipath fading environment in the absence of any prior knowledge. To achieve this, we will deploy a feedforward neural network (FNN) using two different learning representations. The signal data will be represented with the spectral correlation function (SCF) both as the raw I/Q measurements and the compressed form using principal component analysis (PCA).

5.2.1 Feedforward Neural Network

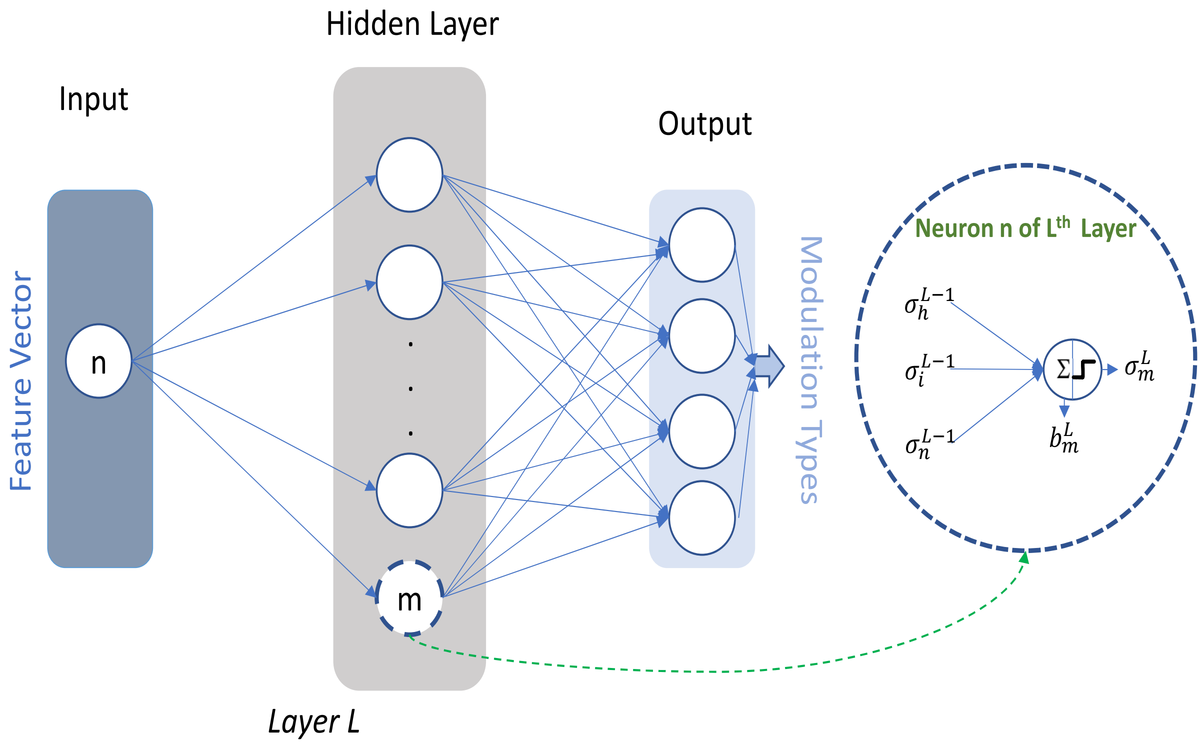


Figure 5.1: Feedforward Neural Network Structure

Feedforward neural networks (FNNs) are extensively used for classification tasks in the literature due to the flexibility of its architecture in providing data-driven solutions for different problems [45]. The general structure of an FNN consists of an input layer, a number of hidden layers, and an output layer, as shown in Figure 5.1. In the first stage of the network, one neuron is assigned to each feature in the dataset to construct the input layer. The hidden layers model the complex non-linear relationship between the input and the output of the system. The final stage includes one neuron for each class type where the number of classes determines the size of the output layer.

Each connection between different neurons in the architecture has an assigned weight, shown as ω. The weight between the two neurons; neuron n in the L^{th} hidden layer and neuron m in the $(L-1)^{st}$ hidden layer, is shown as ω^{L}_{nm}. Hence, one can express the weighted sum at the L^{th} layer as follows:

$$\mathrm{Z}^{\mathrm{L}}_{\mathrm{m}} = \sum_{\mathrm{m}} \omega^{\mathrm{L}}_{\mathrm{nm}}.\, \sigma^{\mathrm{L\text{-}1}}_{\mathrm{n}} + \mathrm{b}^{\mathrm{L}}_{\mathrm{m}}, \tag{5.2}$$

where σ^{L-1}_{n} is the output of the n^{th} neuron in the L^{th} layer and b^{L}_{m} is bias value for each neuron. $\mathrm{Z}^{\mathrm{L}}_{\mathrm{m}}$ is now the input for the next layer after going through an activation function such as the rectified linear unit (ReLU) as follows:

$$f(Z^{L}_{m}) = z^{+} = max(0, z), \tag{5.3}$$

Thus, the output value of the specific neuron m in the L^{th} layer is shown as,

$$\sigma^{L}_{m} = f(z^{L}_{m}), \tag{5.4}$$

For classification purposes, the *softmax* activation function is used in the output layer instead of the ReLU function, σ^{o}_{m}, which is defined as follows:

$$\sigma^{o}_{m} = \frac{e^{z^{L}_{m}}}{\sum_{j} e^{z^{L}_{j}}} \forall m = 1, ..., N, \tag{5.5}$$

where N is the number of neurons in the output layer. In this research, each neuron in the output layer indicates a different modulation type. The intuition behind utilizing the *softmax* function is to assign probabilities to output neurons based on the predicted class label. This is done by one hot encoder, which takes the categorical vectors in the form of a $1xN$ vector which includes $N-1$ zeros and the one index specifies the correct class label. The output values filtered by the *softmax* function in the form of $\sigma^{o}_{m}\epsilon[0, 1]$ represents the probability distributions for all potential outcomes. The neuron with the maximum value in the *softmax* output vector indicates the predicted output label, i.e., the modulation type in this research.

In order to train an FNN, the parameters of the network are updated using an iterative optimization process which aims to minimize a loss function which measures the distance between predictions and actual outputs. In this study, we use the popular "categorical-cross-entropy" which is suitable for multi-class classification problems and is shown below,

$$Loss = -\sum_{j=m}^{M} y_m . log \hat{\sigma}^o_m, \tag{5.6}$$

where $\hat{\sigma}^o_m$ is the m^{th} scalar value in the output layer of size M extracted from the *softmax* function and, y_m is the target value.

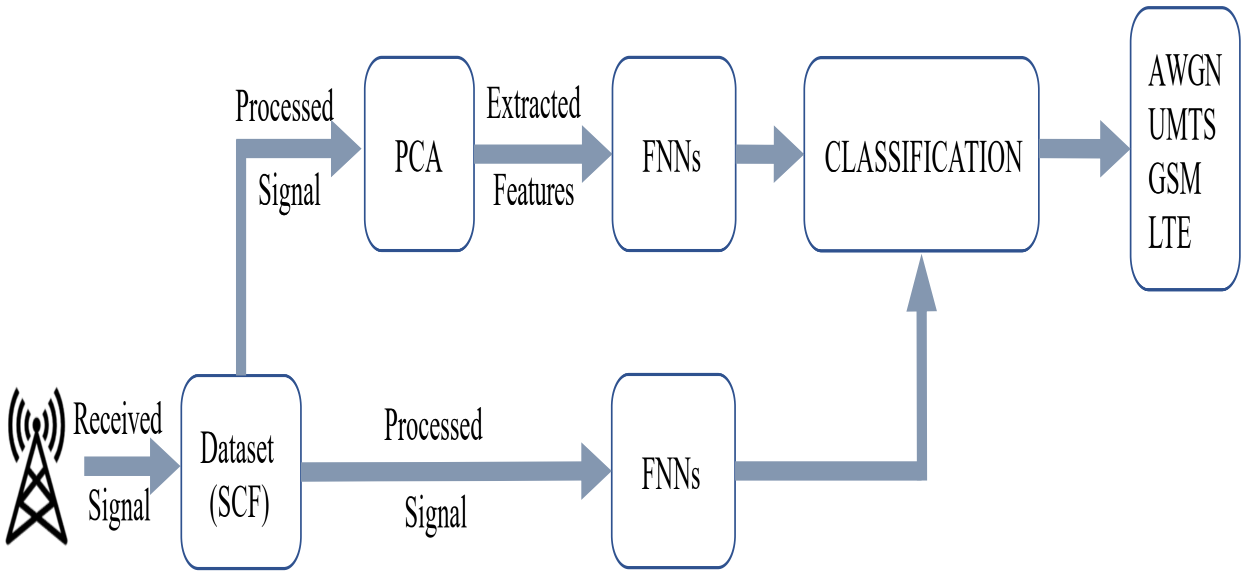


Figure 5.2: Diagram for Proposed Methods

5.2.2 Spectral Correlation Function

In [36], the authors define the cyclostationary signals as ones with periodically changing features (such as the mean and autocorrelation) over time. In other words, cyclostationary signals have hidden periodicities that carry identical characteristics for divergent mobile communication signals such as the symbol periods, code spreading, cyclic prefixes, and even communication data bits. These embedded characteristics generally represent the most relevant information for signal differentiation. The hidden periodicity as a feature of the cyclostationary processes can be extracted from the received signal (1), as mentioned in section II. As with any classification task, one needs to generate an input data matrix with the necessary features and class labels to be able to train the learning algorithm. In this application, the process begins with a non-linear transformation to obtain the second-order cyclostationarity of a signal by autocorrelation as follows,

$$r_\tau(t) = \mathrm{E}\{\mathrm{r}(\mathrm{t} + \tau/2)\mathrm{r}^*(\mathrm{t}\text{-}\tau/2)\}, \tag{5.7}$$

where $r_\tau(t)$ is the autocorrelation function of $r(t)$. Considering the autocorrelation function is periodic with T_0, it's Fourier series coefficients can be calculated as follows:

$$R_r^\alpha(\tau) = \frac{1}{T_0}\int_{-T_0/2}^{T_0/2} r_\tau(t)e^{-j2\pi\alpha t}dt, \tag{5.8}$$

where R_τ^α is called the cyclic autocorrelation function (CAF) and the frequencies $\alpha = n/T_0$, $n \epsilon \mathrm{Z}$ are called the cyclic frequencies.

The Fourier transform of the cyclic autocorrelation function at a cyclic frequency is called the spectral correlation function (SCF) which is expressed as,

$$\mathrm{S}_\mathrm{r}^\alpha(f) = \int_{-T/2}^{T/2} \mathrm{R}_\mathrm{r}^\alpha(\tau)e^{-j2\pi ft}d\tau, \tag{5.9}$$

where $\mathrm{S}_r^\alpha(f)$ is the same as the average power spectrum density (PSD) when alpha is equal to zero.

Calculating the SCF is a complex process. However, Fast Fourier Transform (FFT) accumulation method (FAM) [82] helps decrease the computational complexity. Ultimately, for each received signal with modulations GSM, UMTS or LTE, the input matrix can be constructed as follows:

$$\mathbf{X}_k^{SCF} = |\mathsf{S}_{r\tau}\ (nL, f)|, \tag{5.10}$$

where $\mathsf{S}_{r\tau}\ (nL, f)$ is the FAM approximated SCF and L is the length of the FFT.

The mathematical processes between equations (5.9) and (5.10) are omitted in this text for the sake of brevity, however the reader is referred to [92] for a detailed discussion on this subject. The input matrix shown in (5.10) is used as magnitudes since the complex values are not supported by the Python environment for general classification tasks.

5.3 Experimental Setup

5.3.1 Dataset

In this study, the dataset first introduced by [92] is used which considers real-world scenarios based on actual measurements in an urban area. The data is represented in the form of the Spectral Correlation Function (SCF) [94] which has previously been shown to provide the best performance for classification accuracy among other formats such as fusion in [92]. The dataset includes three different signal types stemming from the Global System for Mobile Communications (GSM), wideband code division multiple access (WCDMA) for the universal mobile telecommunications system (UMTS), and long-term evolution (LTE).

For the measurements, different locations and bands have been considered. In the experiment, transmitters are placed in the urban areas, but receivers are placed in the sub-urban areas. Thus, the signals propagate from the urban area directly whereas the receiver collects them in the sub-urban area. Evaluation in this dataset was done by focusing on the 800, 900, 1800, and 2100 MHz bands, which cover the entire cellular communication spectrum. Those bands represent different classes at 15 different signal-to-noise ratio (SNR) levels. For each class under each noise level, a total of 1000 samples are collected with 60% of them being used as the training data and the remaining 40% as the testing data. Each signal has 16384 in-phase/quadrature (I/Q) samples such that the size of the observation matrix for each signal is 8193x16 – a significant size compared to more conventional classification problems.

The signals in the dataset have specific characteristics based on the environment in which they were generated and/or measured. For example, the received signal is affected by different wireless spreading channels such as multipath fading, shadowing, and path attenuation in the measurement area. Therefore, the power and phase of the signal have different scales. Moreover, the received signal has different power distributions since it is recorded at different receivers and locations within identical characteristic bands. Finally, phase spreading characteristics are uniform between $-\pi$ and π, similar to Rayleigh fading channel properties. As the authors indicate in [92], the bandwidth and carrier frequency are not represented in the dataset to allow for the cognitive radios to perform at every point on the spectrum opportunistically.

5.3.2 Performance Metric

A commonly used evaluation metric for multiclass classification problems is the average classification accuracy. The goal is to look at the mean accuracy on the test data which can be defined simply as follows,

$$Accuracy(y, \hat{y}_j) = \frac{1}{n_{samples}} \sum_{j=0}^{n_{samples}-1} (\hat{y}_j == y_j), \tag{5.11}$$

where $\hat{y}_j$ is the predicted value of the j^{th} sample and y_j is the true value.

5.3.3 Implementation

All the software implementations are done using the Keras library [19] on Spyder, as a popular scientific Python environment for the training and testing processes. We investigate the performance of two models where the data is represented in two fundamentally different ways. In the first case, an FNN with the D-100-4 structure where D represents the size of the feature space is trained using the raw I/Q samples such that the dimension of the input layer is 131088 (8193 x 16). In the second case, PCA is applied to reduce the dimensionality of the input space before training the FNN. The variance margin of the data is set at 85% for the PCA which creates a significantly smaller feature set for training.

The training and testing partitions of the dataset are set as 60% and 40%, respectively. Hence, 2400 samples were separated for training and 1600 for testing for each SNR value. In training, early stopping is employed to avoid overfitting. Adam optimizer is used with 600 epochs for each SNR level with a 25% validation split used in early stopping. For statistically meaningful results, the initialization of the network is randomized 20 times and the average accuracy across all the trials are reported. Both models using raw data and PCA were trained and on a workstation computer with an AMD RYZEN 5800@3.2 GHz X 8 CPU, 16GB RAM, and an NVIDIA GeForce RTX 3070 GPU. The training and testing times have been recorded for different learning scenarios at 5dB and 15dB to compare the computational time with corresponding classification performance.

5.4 Results and Discussions

The classification performance of the proposed methods including the raw and PCA representations of the data are evaluated over a real dataset of wireless communications signals, and their powers of signal identification (classification) for different SNR levels from 1 to 15 dB are compared to the current state-of-the-art and represented in Table 5.1 and Figures 5.3 through 5.4. In this chapter, the signal identification power of the proposed models are indicated by their average classification accuracies.

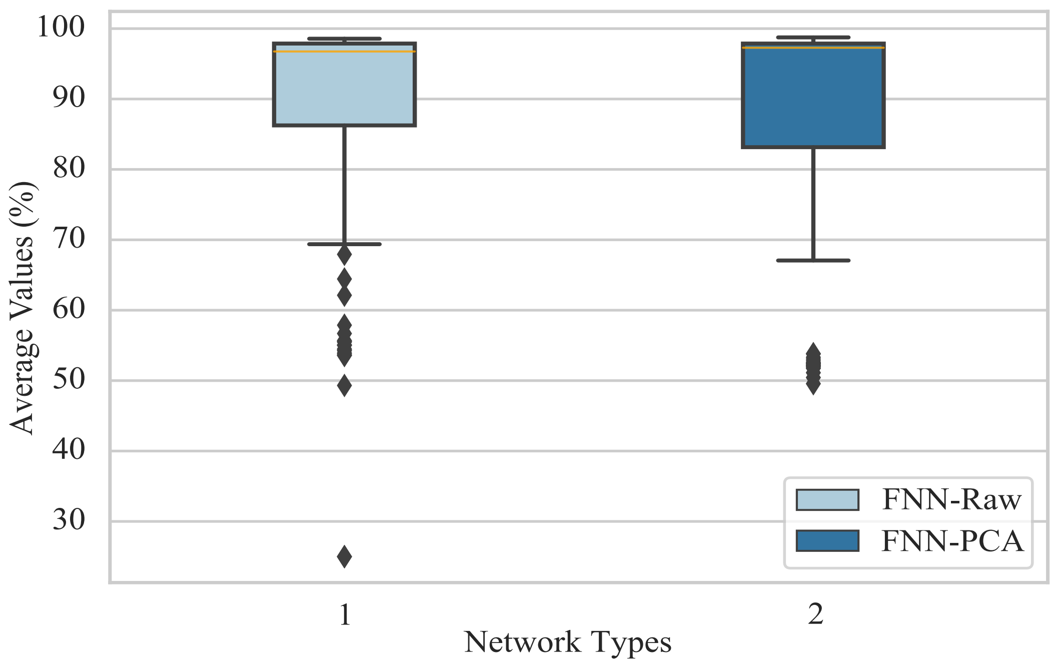


Figure 5.3: Box Plots for Average Classification Accuracy Includes All 300 Statistical Values.

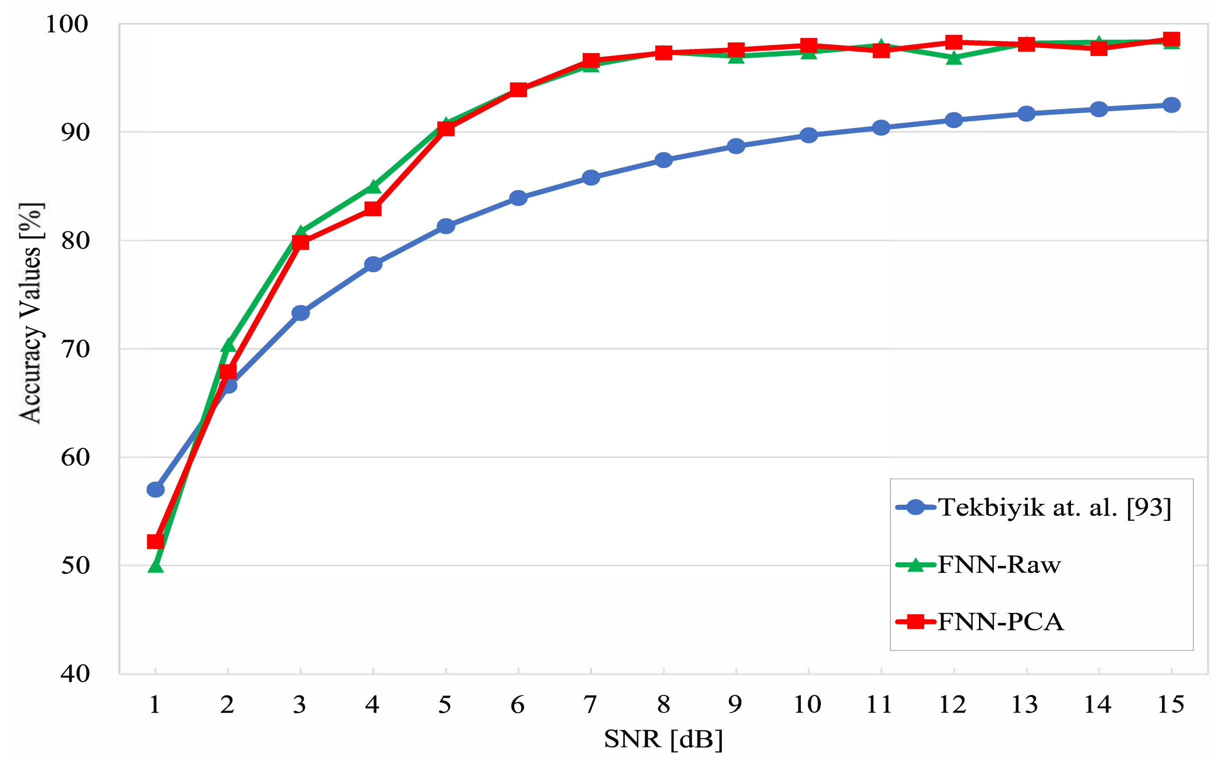


Figure 5.4: Average Accuracies Values for FNN-Raw and FNN-PCA.

Table 5.1: Classification Accuracy for SNR Values in Between 1 to 15 dB.

AVERAGE CLASSIFICATION ACCURACY (%)			
SNR/dB	Tekbiyik et. al.[93]	FNNs	Applied PCA (Component %85)
1	57	50	52.2
2	66.6	70.4	67.9
3	73.3	80.8	79.8
4	77.8	85	82.9
5	81.3	90.8	90.3
6	83.9	93.9	93.9
7	85.8	96.2	96.6
8	87.4	97.4	97.3
9	88.7	97	97.6
10	89.7	97.4	98
11	90.4	98	97.5
12	91.1	96.9	98.3
13	91.7	98.2	98.1
14	92.1	98.3	97.7
15	92.5	98.3	98.6

In Table 5.1, the following observations can be made. At the lowest SNR level, the test accuracy levels exceed 50% for FNN-Raw and 52% for FNN-PCA, which means that the proposed methods can still distinguish signals to some extent, even at an extremely low SNR as the original study claims that no signal exists at this level. One can observe that all models perform similarly at high SNR values (around 98%) whereas the performance of the proposed models begin to significantly outperform the previously reported results around SNR = 11dB with an average performance difference of 4-5% between models.

Figure 5.3 shows the distributions of model accuracies across all SNR levels as box-plots. Since there are 15 different SNR levels and 20 repetitions at each SNR levels, this figure represents 300 accuracy values for each of the proposed models. One can observe that while both FNN-Raw and FNN-PCA perform similarly at the mean, there is a bigger variance of accuracies for the PCA model. This is not entirely unexpected as the PCA model compresses the feature space likely losing some of the important outlier information, especially at low SNR values, which can improve the performance of the raw data model. However, when considering the training and execution times, the PCA model significantly outperforms the raw model where the average training time per epoch is 1.2 seconds for FNN-Raw and only 0.0003 seconds for FNN-PCA. More importantly, the execution time (or testing time) for the PCA model is only 0.06 seconds compared to 3.22 seconds (almost a 50-fold increase) for the raw data model. Execution time is critical for signal identification where switching systems can quickly adapt to the modulation if the signal is identified both accurately and quickly. It is important to note that the training times are reported per epoch whereas testing times are reported over the entire test set for a fair comparison with previous state-of-the-art. Finally, it is worth mentioning that the PCA model is competitive at higher SNR levels which means that when the channel is controlled, the execution time and accuracy metrics both work in favor of the compressed learning representation.

Chapter 6: Conclusions and Future Work

Chapter 3 presents a comprehensive study in deep unsupervised learning to observe the effect of hyper-parameters including network topology and clustering coefficients on deep clustering accuracy. Popular autoencoder and convolutional autoencoder architectures are used to obtain clustering friendly features in the latent space where a standard k-means back-end algorithm implements the clustering. The experimental results show that the hyper-parameters influence clustering accuracy performance in both expected and unexpected ways. For instance, more complex datasets require a higher dimensional latent space to achieve the best subsequent clustering performance. For the image datasets, color can help improve the clustering performance but only when it signifies relevant information on the identity of the observation. Additional studies need to be conducted to truly understand the complex relationship between the hyperparameters, network topologies and the statistics and complexities of the datasets when it comes to deep clustering applications.

In chapter 4, we explored the premise of conventional machine learning when compared to deep learning for anomaly detection in SONs. Anomaly detection was a popular application area of deep learning for cell outages in communication networks. However, as in other domains, conventional methods can still provide strong statistical alternatives to the right learning representations. In this research, we focused on SVMs with one-class and binary learning scenarios on a previously published and publicly available dataset. We found that while deep learning was highly competitive, standard SVMs using RBF kernels, can be trained to outperform a deep autoencoder approach. Both one-class and binary classification can benefit immensely from synthetic augmentation of the dataset using SMOTE with improvements in detection accuracy by as much as 15% on average over four different application scenarios.

Future work will study the impact of augmentation on other learning algorithms, specifically statistical deep learning, such as variational auto-encoders. Work presented in this paper can further be extended to other applications beyond anomaly or outage detection. Specifically, there has been increased attention to modulation detection in next generation mobile wireless networks where fast, robust, and light machine learning models could enable time-critical applications in signal classification and modulation detection. Improvements in speed can be realized both at the algorithm level and data preprocessing stages using techniques such as principal component analysis to identify the most relevant features for classification and detection. Finally, statistical learning algorithms, such as Gaussian Process Regression, which have gained immense popularity as alternatives to deep learning can be applied to different scenarios especially when data is not present in sufficiently large volumes to properly train DL models with many parameters.

In chapter 5, we present a comparative study of a lightweight conventional machine learning algorithm with and without data compression for modulation detection on a large real-world dataset. The latest report on this dataset published in the IEEE Transactions on Vehicular Technology demonstrated the promise of data-centric deep learning which can perform robust signal classification at a variety of noise levels. The proposed algorithm outperforms the current state-of-the-art by as much as 6% on average across different noise levels from 1dB to 15dB SNR. We observe that the true gains however are in execution times where the detection speed is improved by at least 50-fold keeping approximately the same median accuracy with only a marginal increase in error variance. Future work will implement additional reductions in computational complexity while maintaining similar performance levels.

References

[1] Airhop. Accessed on: June 18, 2021. [Online]. Available: http://www.airhopcomm.com.

[2] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Transactions on Network and Service Management*, 16(2):445–458, 2019.

[3] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, Maximilian Strobel, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018.

[4] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[5] Jeffrey G Andrews, Stefano Buzzi, Wan Choi, Stephen V Hanly, Angel Lozano, Anthony CK Soong, and Jianzhong Charlie Zhang. What will 5g be? *IEEE Journal on selected areas in communications*, 32(6):1065–1082, 2014.

[6] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[7] Ahmad Asghar, Hasan Farooq, and Ali Imran. Self-healing in emerging cellular networks: Review, challenges, and research directions. *IEEE Communications Surveys & Tutorials*, 20(3):1682–1709, 2018.

[8] Muhammad Zeeshan Asghar. Zeeshan: Simulation data sets.

[9] Muhammad Zeeshan Asghar, Mudassar Abbas, Khaula Zeeshan, Pyry Kotilainen, and Timo Hämäläinen. Assessment of deep learning methodology for self-organizing 5g networks. *Applied Sciences*, 9(15):2975, 2019.

[10] Arian Azarang, Hafez E Manoochehri, and Nasser Kehtarnavaz. Convolutional autoencoder-based multispectral image fusion. *IEEE access*, 7:35673–35683, 2019.

[11] Alexandru-Sabin Bana, Elisabeth De Carvalho, Beatriz Soret, Taufik Abrao, José Carlos Marinello, Erik G Larsson, and Petar Popovski. Massive mimo for internet of things (iot) connectivity. *Physical Communication*, 37:100859, 2019.

[12] Christopher M Bishop. *Pattern recognition and machine learning.* springer, 2006.

[13] Federico Boccardi, Robert W Heath, Angel Lozano, Thomas L Marzetta, and Petar Popovski. Five disruptive technology directions for 5g. *IEEE communications magazine*, 52(2):74–80, 2014.

[14] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[15] Jesús Burgueño, Isabel de-la Bandera, Jessica Mendoza, David Palacios, Cesar Morillas, and Raquel Barco. Online anomaly detection system for mobile networks. *Sensors*, 20(24):7232, 2020.

[16] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[17] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. Autoencoder-based network anomaly detection. In *2018 Wireless Telecommunications Symposium (WTS)*, pages 1–5. IEEE, 2018.

[18] François Chollet et al. Keras, 2015.

[19] François Chollet. keras. https://github.com/fchollet/keras, 2015.

[20] Domenico Ciuonzo, Pierluigi Salvo Rossi, and Subhrakanti Dey. Massive mimo channel-aware decision fusion. *IEEE Transactions on Signal Processing*, 63(3):604–619, 2014.

[21] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[22] DL Davies and DW Bouldin. A cluster separation measure, ieee transactions on patter analysis and machine intelligence. vol, 1979.

[23] Isabel de-la Bandera, Raquel Barco, P Munoz, and Inmaculada Serrano. Cell outage detection based on handover statistics. *IEEE Communications Letters*, 19(7):1189–1192, 2015.

[24] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[25] Octavia A Dobre. Signal identification for emerging intelligent radios: Classical problems and new challenges. *IEEE Instrumentation & Measurement Magazine*, 18(2):11–18, 2015.

[26] Octavia A Dobre, Ali Abdi, Yeheskel Bar-Ness, and Wei Su. The classification of joint analog and digital modulations. In *MILCOM 2005-2005 IEEE Military Communications Conference*, pages 3010–3015. ieee, 2005.

[27] Octavia A. Dobre and Fahed Hameed. Likelihood-based algorithms for linear digital modulation classification in fading channels. In *2006 Canadian Conference on Electrical and Computer Engineering*, pages 1347–1350, 2006.

[28] Joseph C Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1):95–104, 1974.

[29] Yahia A Eldemerdash, Octavia A Dobre, Oktay Üreten, and Trevor Yensen. Identification of cellular networks for intelligent radio measurements. *IEEE Transactions on Instrumentation and Measurement*, 66(8):2204–2211, 2017.

[30] Ericson. Network outage. Accessed on: June 18, 2021.

[31] A Fehske, J Gaeddert, and Jeffrey H Reed. A new approach to signal classification using spectral correlation and neural networks. In *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.*, pages 144–150. IEEE, 2005.

[32] Hasna Fourati, Rihab Maaloul, Lamia Chaari, and Mohamed Jmaiel. Comprehensive survey on self-organizing cellular network approaches applied to 5g networks. *Computer Networks*, 199:108435, 2021.

[33] Edward B Fowlkes and Colin L Mallows. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569, 1983.

[34] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[35] William A Gardner. Exploitation of spectral redundancy in cyclostationary signals. *IEEE Signal processing magazine*, 8(2):14–36, 1991.

[36] William A Gardner and Enders A Robinson. Statistical spectral analysis—a nonprobabilistic theory. 1989.

[37] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5736–5745, 2017.

[38] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[39] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International Conference on Neural Information Processing*, pages 373–382. Springer, 2017.

[40] Wuri A Hapsari, Anil Umesh, Mikio Iwamura, Malgorzata Tomala, Bódog Gyula, and Benoist Sebire. Minimization of drive tests solution in 3gpp. *IEEE Communications Magazine*, 50(6):28–36, 2012.

[41] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[42] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[43] Chih-Chung Hsu and Chia-Wen Lin. Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data. *IEEE Transactions on Multimedia*, 20(2):421–429, 2017.

[44] Kazi Mohammed Saidul Huq, Sherif Adeshina Busari, Jonathan Rodriguez, Valerio Frascolla, Wael Bazzi, and Douglas C. Sicker. Terahertz-enabled wireless system for beyond-5g ultra-fast networks: A brief survey. *IEEE Network*, 33(4):89–95, 2019.

[45] Jithin Jagannath, Nicholas Polosky, Daniel O'Connor, Lakshmi N Theagarajan, Brendan Sheaffer, Svetlana Foulke, and Pramod K Varshney. Artificial neural network based automatic modulation classification over a software defined radio testbed. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[46] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[47] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

[48] Johan Johansson, Wuri A Hapsari, Sean Kelley, and Gyula Bodog. Minimization of drive tests in 3gpp release 11. *IEEE Communications Magazine*, 50(11):36–43, 2012.

[49] Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

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

[51] Ria Kanjilal and Ismail Uysal. The future of human activity recognition: Deep learning or feature engineering? *Neural Processing Letters*, 53(1):561–579, 2021.

[52] Ebrahim Karami, Octavia A Dobre, and Nikhil Adnani. Identification of gsm and lte signals using their second-order cyclostationarity. In *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, pages 1108–1112. IEEE, 2015.

[53] Ilyas Alper Karatepe and Engin Zeydan. Anomaly detection in cellular network data using big data analytics. In *European Wireless 2014; 20th European Wireless Conference*, pages 1–5. VDE, 2014.

[54] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[56] Muhammed Kucuk and Ismail Uysal. Performance analysis of neural network topologies and hyperparameters for deep clustering. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2020.

[57] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 12 1989.

[58] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[59] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[60] Fengfu Li, Hong Qiao, and Bo Zhang. Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognition*, 83:161–173, 2018.

[61] Wenwen Li, Zheng Dou, Lin Qi, and Chengzhuo Shi. Wavelet transform based modulation classification for 5g and uav communication in multipath fading channel. *Physical Communication*, 34:272–282, 2019.

[62] Xiang Lin, Octavia A Dobre, Telex MN Ngatched, Yahia A Eldemerdash, and Cheng Li. Joint modulation classification and osnr estimation enabled by support vector machine. *IEEE Photonics Technology Letters*, 30(24):2127–2130, 2018.

[63] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[64] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[65] Jessica Moysen, Furqan Ahmed, Mario García-Lozano, and Jarno Niëmela. Unsupervised learning for detection of mobility related anomalies in commercial lte networks. In *2020 European Conference on Networks and Communications (EuCNC)*, pages 111–115. IEEE, 2020.

[66] Abbass Nasser, Ali Mansour, K-C Yao, Hani Abdallah, Mohamad Chaitou, and Hussein Charara. Spectrum sensing enhancement using principal component analysis. In *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 263–267. IEEE, 2016.

[67] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[68] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[69] Timothy O'shea and Jakob Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.

[70] Timothy James O'Shea, Tamoghna Roy, and T Charles Clancy. Over-the-air deep learning based radio signal classification. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):168–179, 2018.

[71] Sankar K Pal and Sushmita Mitra. Multilayer perceptron, fuzzy sets, classifiaction. 1992.

[72] VASILE Palade. Class imbalance learning methods for support vector machines. *Imbalanced Learning: Foundations, Algorithms, and Applications; Wiley: Hoboken, NJ, USA*, page 83, 2013.

[73] Xi Peng, Ivor W Tsang, Joey Tianyi Zhou, and Hongyuan Zhu. k-meansnet: When k-means meets differentiable programming. *arXiv preprint arXiv:1808.07292*, 2018.

[74] Jide Julius Popoola and Rex Van Olst. A novel modulation-sensing method. *IEEE Vehicular Technology Magazine*, 6(3):60–69, 2011.

[75] Wencong Qin, Yinglei Teng, Mei Song, Yinghai Zhang, and Xiaojun Wang. Aq-learning approach for mobility robustness optimization in lte-son. In *2013 15th IEEE International Conference on Communication Technology*, pages 818–822. IEEE, 2013.

[76] Sreeraj Rajendran, Wannes Meert, Vincent Lenders, and Sofie Pollin. Unsupervised wireless spectrum anomaly detection with interpretable features. *IEEE Transactions on Cognitive Communications and Networking*, 5(3):637–647, 2019.

[77] Sharan Ramjee, Shengtai Ju, Diyu Yang, Xiaoyu Liu, Aly El Gamal, and Yonina C Eldar. Fast deep learning for automatic modulation classification. *arXiv preprint arXiv:1901.05850*, 2019.

[78] Barathram Ramkumar. Automatic modulation classification for cognitive radios using cyclic feature detection. *IEEE Circuits and Systems Magazine*, 9(2):27–45, 2009.

[79] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[80] Raimundo Real and Juan M Vargas. The probabilistic basis of jaccard's index of similarity. *Systematic biology*, 45(3):380–385, 1996.

[81] Douglas Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, pages 827–832, 2015.

[82] Randy S Roberts, William A Brown, and Herschel H Loomis. Computationally efficient algorithms for cyclic spectral analysis. *IEEE Signal Processing Magazine*, 8(2):38–49, 1991.

[83] Marko M Roganovic, Aleksandar M Neskovic, and Natasa J Neskovic. Application of artificial neural networks in classification of digital modulations for software defined radio. In *IEEE EUROCON 2009*, pages 1700–1706. IEEE, 2009.

[84] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[85] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[86] Mark Sanderson. Christopher d. manning, prabhakar raghavan, hinrich schütze, introduction to information retrieval, cambridge university press. 2008. isbn-13 978-0-521-86571-5, xxi 482 pages. *Natural Language Engineering*, 16(1):100–103, 2010.

[87] Yasuyosi Sasaki. Power control unit for high power hybrid system. In *SAE2007World Congress*, pages 1–5, 2007.

[88] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[89] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[90] Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Iberoamerican Congress on Pattern Recognition*, pages 117–124. Springer, 2013.

[91] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[92] Kürşat Tekbiyik, Özkan Akbunar, Ali Riza Ekti, Ali Görçin, and Güneş Karabulut Kurt. Multi–dimensional wireless signal identification based on support vector machines. *IEEE Access*, 7:138890–138903, 2019.

[93] Kürşat Tekbıyık, Özkan Akbunar, Ali Rıza Ekti, Ali Görçin, Güneş Karabulut Kurt, and Khalid A Qaraqe. Spectrum sensing and signal identification with deep learning based on spectral correlation function. *IEEE Transactions on Vehicular Technology*, 70(10):10514–10527, 2021.

[94] Kürşat TEKBIYIK, Özkan Akbunar, Ali Rıza Ekti, Ali Görçin, and Güneş Karabulut Kurt. Cosine: Cellular communication signal dataset, 2020.

[95] James T Townsend. Theoretical analysis of an alphabetic confusion matrix. *Perception & Psychophysics*, 9(1):40–50, 1971.

[96] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[97] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[98] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010.

[99] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001.

[100] Song Wang, Juan Fernando Balarezo, Sithamparanathan Kandeepan, Akram Al-Hourani, Karina Gomez Chavez, and Benjamin Rubinstein. Machine learning in network anomaly detection: A survey. *IEEE Access*, 9:152379–152396, 2021.

[101] Alan Weissberger. Nokia and vodafone to use machine learning on google cloud to detect network anomalies, Jul 2021.

[102] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016.

[103] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3861–3870. JMLR. org, 2017.

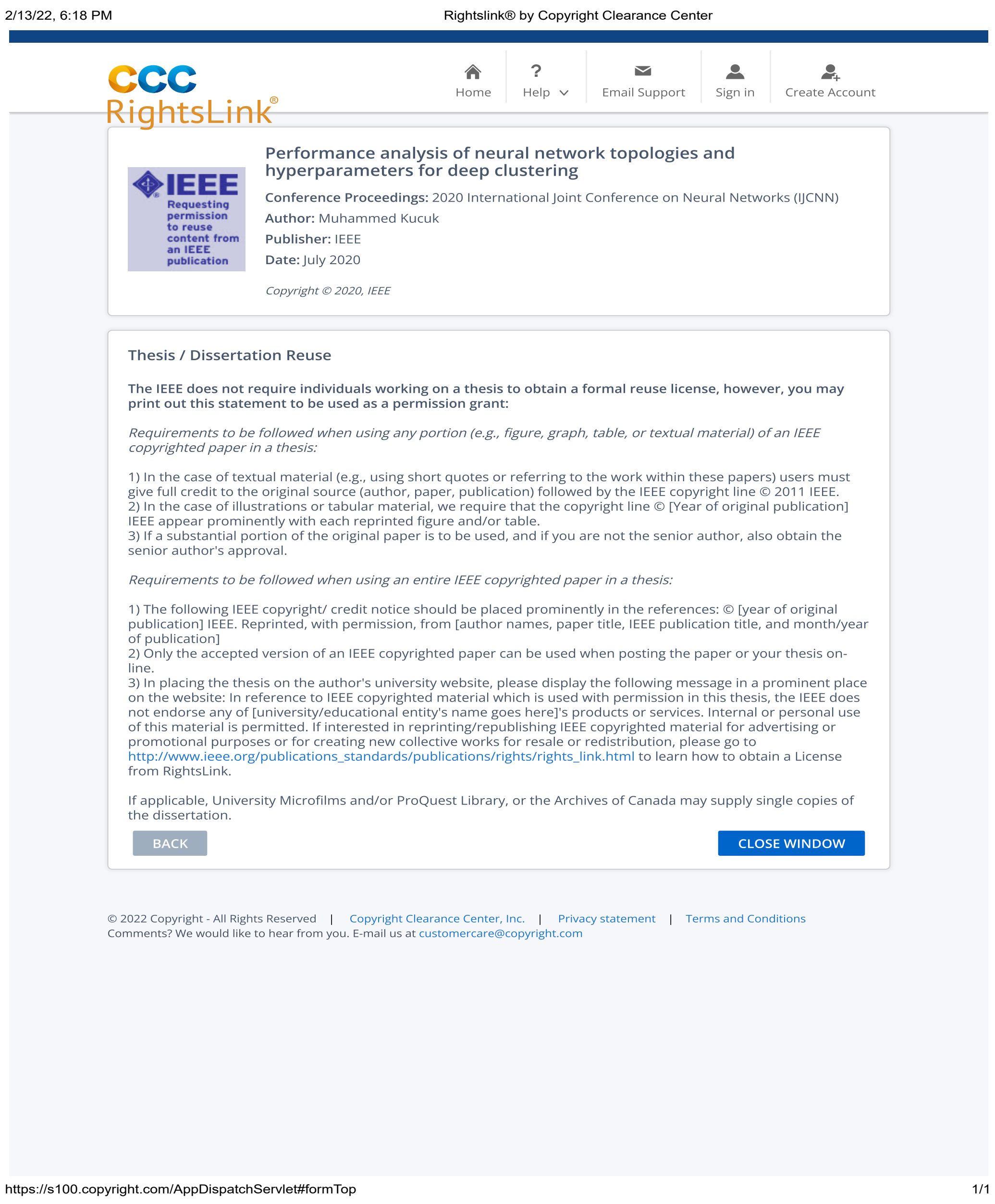
[104] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

[105] Jaehyun Yoo, Karl Henrik Johansson, and Hyoun Jin Kim. Indoor localization without a prior map by trajectory learning from crowdsourced measurements. *IEEE Transactions on Instrumentation and Measurement*, 66(11):2825–2835, 2017.

[106] Ahmed Zoha, Arsalan Saeed, Ali Imran, Muhammad Ali Imran, and Adnan Abu-Dayya. Data-driven analytics for automated cell outage detection in self-organizing networks. In *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*, pages 203–210. IEEE, 2015.

Appendix A: Copyright Permissions

The permission below is for the reproduction of material in Chapter 1 and 3.



2/13/22, 6:18 PM Rightslink® by Copyright Clearance Center

CCC RightsLink®
Home
Help
Email Support
Sign in
Create Account
IEEE Requesting permission to reuse content from an IEEE publication

Performance analysis of neural network topologies and hyperparameters for deep clustering

Conference Proceedings: 2020 International Joint Conference on Neural Networks (IJCNN)

Author: Muhammed Kucuk

Publisher: IEEE

Date: July 2020

Copyright © 2020, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

https://s100.copyright.com/AppDispatchServlet#formTop 1/1

About the Author

Muhammed Furkan Kucuk (SM'18) received his B.S. degree in electrical and electronics engineering from the University of Gaziantep, Gaziantep, Turkey, in 2014, and his M.S. degree in electrical engineering from the University of South Florida (USF), Tampa, FL, USA, in 2018. He is currently pursuing his Ph.D. degree at USF in the same department.

Since 2018, he has been a Research Assistant with USF. His research interests include machine learning, deep neural networks for unsupervised learning and ML applications in wireless communications. His award and honors include the Turkish Government Scholarship for M.S. and Ph.D. degrees.