**ABSTRACT**

*Credit card fraud is a prevailing issue that has now become a risk to the economy. It is a growing concern that is increasing in severity with the popularity of internet. Nowadays, e-commerce, online shopping, and other internet-based transactions has made credit card fraud much more common large due to inadequate security measures taken by financial institutions, vendors or dealers, and individuals themselves. There are many credit card fraud detection systems currently in place, but these are ineffective mainly because these are inflexible while techniques used for fraud are adaptive and flexible. Thus, there is a need for a fraud detection system that is not only highly accurate but adaptive to changing fraud techniques as well. In this paper, we present a neural-network based fraud detection system for identifying and preventing fraudulent transactions. The aim of this paper is to present an accurate, flexible, and reliable system that can be used for classifying transactions in real-time as they are being processed rather than after they have occurred to maximize the chances of prevention and minimize the losses that occur due to fraudulent transactions. In addition to this, this paper aims to contribute to the exiting literature by testing a fraud detection system implemented in the Python-framework Keras with the multi-layer perceptron architecture for the neural network, ReLU and Sigmoid activation functions, the Adam optimizer, and a cost-sensitive function for computing accuracy. The dataset that has been used for this paper is a PaySim mobile payment dataset acquired from Kaggle.*

**CHAPTER 1: INTRODUCTION**

Online fraud is an issue that has been around ever since the internet became popular. It is one of the most common problems in the digital world and it leads to a loss of millions of dollars’ worth of money to businesses, financial institutes, and individuals. We define fraud as any act or deception that is intended for the financial or personal gain of the fraudster. Online fraud is one such kind of fraud which is carried out through digitally (or through the internet). Credit card fraud is defined as the theft of the sensitive credentials of a payment card such as a credit card or debit card, which is used illegitimately for making a transaction.

The reason we choose this particular type of fraud for our project is because it is the most practical approach to monitor ‘real-time’ online transactions which can be prevented and reversed within due time. As opposed to this, analyzing offline transactions cannot always guarantee reversal or prevention. Furthermore, credit card fraud is a rising problem that is yet to be solved. Our goal is to experiment and present a solution that can reduce, if not eliminate, the risk of credit card fraud.

Credit card fraud has been around since a long time, but the issue has grown substantially over time. In 2013, the number of data breaches were 614 while this number was projected to increase up to a considerable 1579 in 2017. [1] The total monetary loss which parties incurred due to credit card not present (CNP) fraud is expected to reach $6.4 billion in 2018 in the US alone, which itself is a massive three times rise from 2016. [2]

These statistics show how the problem of fraudulent credit card transactions has been increasing in recent years. In the past, before this digital age, credit cards had to be physically stolen, duplicated, or otherwise acquired to carry out a fraud transaction. However, now with the emergence of electronic payments, people can use your credit card by just knowing the card number and security code which can easily be acquired through methods such as phishing, hacking and data breaches, and social engineering among others.

One of the primary concerns associated with credit card fraud is to identify and hence, reverse fraudulent transactions in a timely manner. Since the ratio of fraudulent credit card transactions to real credit card transactions is very small, it is extremely difficult to identify fraud transactions out of millions of real transactions.

The main aim of this project is to propose a methodology through concepts of machine learning and data science to efficiently and accurately identify a fraudulent credit card transaction.

**What is a Neural Network?**

A neural network, more formally known as an artificial neural network, is a machine learning model which based on the structure and functioning of biological neural networks. A neural network can be developed through supervised learning, unsupervised learning, and reinforcement learning. Typically, neural networks are organized in a number of layers which are classified as the *Input Layer*, *Hidden Layer(s)*, and *Output Layer*. Each of these layers is described below:

1. **Input Layer:** This is the layer that deals with the external environment to feed the input into the neural network model. Each independent attribute (variable) in the dataset which has an impact on the output is a neuron in the input layer.
2. **Hidden Layer(s):** This is the layer in which the inputs are processed. It is an intermediary layer between the input layer and output layer in which an activation function, such as the sigmoid function, is applied to process the data fed to the hidden layer from the previous layer. Theoretically, this is the layer which is responsible for extracting the meaning and required features from the input layer. It is important to mention that there can be more than one hidden layer in a neural network, depending on the requirements and complexity of the dataset. In each hidden layer, the number of neurons can vary. There is no definite formula (or method) known to determine the exact number of hidden layers or neurons per hidden layer in a neural network.
3. **Output Layer:** This is the layer where the processed data is transmitted from the neural network. The number of neurons in the output layer depend on the type of analysis that the neural network is performing and it can vary depending on the desired outcome from the neural network.

In a neural network, the communication between the various layers is carried out through connections between the neurons. Each connection has a weight assigned to it which is typically assigned randomly in the beginning but is then modified and *improved* through a learning rule. An example of a commonly used learning rule is the learning rule in which the weights are recalculated through the backward propagation of the error in the output value. A neural network in which backpropagation occurs is known as a backpropagation-al neural network (BPNN).

**Artificial Neural Network (Image):** [https://cdn-images-1.medium.com/max/1600/1\*hczvrCYgU\_JQt5sx-UGM1A.gif](https://cdn-images-1.medium.com/max/1600/1*hczvrCYgU_JQt5sx-UGM1A.gif)

As with other machine learning models, a neural network is first trained using a part of the dataset which is referred to as the training dataset. To train the neural network model, the weights are updated several times by applying a learning rule until the weights and outputs are accurate to a satisfactory level.

There are numerous applications of neural networks which include the recognition of patterns or discovery of regularities within a dataset. Generally, a neural network performs very well when the dataset is very diverse, complex, or involves a high number of variables. It is best suited for problems where the relationship between variables is vague and dynamic (non-linear) in nature.

**What is Keras?**

Keras is an open source machine learning library which has been written in Python. The library is widely used to model neural networks and enable experimentation on neural networks. Keras is commonly used as a wrapper to Tensorflow. It is highly popular because of it is modular, extensible, and user-friendly.

The library contains a number of implementations of all the basic building blocks of a neural network. These include the layers, activation functions, learning rules, optimizers, and objectives in addition to a number of utilities for handling text-based and image-based data. Through Keras, neural network models can be developed on the web, on a Java Virtual Machine (JVM), and smartphones (such as iOS and Android).

The benefit of using Keras is that it simplifies the process of building a neural network. Without Keras, individual components of the neural network have to be developed from scratch and then have to be adjusted according to the problem. Keras enables us to work directly on the neural network and hence making the code simpler, efficient, and cleaner to write, maintain, and troubleshoot.

Here are the advantages of using the Keras machine learning library:

1. Neural network models developed on Keras can be easily transformed into products that can be released on a range of platforms such as mobile and web.
2. There is support for a number of backend engines compatible with Keras which include the Tensorflow backend and CNTK backend.
3. It is developer-centric and not machine-centric. It is easy to use and flexible at the same time.
4. The research community and industry are widely using Keras and it is supported by key companies such as Google and Microsoft.

In the next chapter, we present the studies and solutions that have been proposed by researchers in literature for solving the issue of credit card fraud. We critically evaluate and utilize this research for then describing and proposing a machine learning based solution for the detection of fraudulent credit card transactions.

**Aims:**

The aim of this project is to develop a system that can monitor, detect, and prevent fraudulent credit card transactions. As mentioned in the problem statement, credit card fraud is a challenging problem for financial institutions and there is a need for a centralized generalized solution that can resolve this issue. There are several current systems and methodologies in place for detecting credit card fraud. We aim to improve these systems by using machine learning and data science. Through the use of these concepts, we aim to develop an automated and intelligent fraud transaction detection system which is highly accurate, reliable, and cost-effective.

The existing systems that are used for credit card fraud detection are expert systems which make use of patterns and pre-defined rules for classifying transactions as either fraudulent or legitimate. Though this was an effective approach in the past, it is not very scalable or accurate when there is a very large number of transactions. Furthermore, a known issue with these systems is that they report a high number of false positives i.e. high False Acceptance Ratio (FAR). Ever since the introduction of online payments, credit card fraud has been on an alarming rise and thus there is an increasing need for an intelligent and accurate credit card fraud detection system. Rather than using pre-defined uses which can become obsolete, we aim to make use of machine learning that can enable the fraud detection system to gather information and make rules for classification on its own. This enables the system to produce results that are precise, accurate, and consistent.

Consequently, the aim of this project is to analyze, understand, and apply concepts of machine learning for transaction management, in general, and credit card fraud detection, in specific. This this, we aim to identify existing literature on the subject matter and make use of the methodologies applies and proposed in the literature to develop a machine learning based fraud transaction detection system. Our aim is to develop a system that is able to detect fraudulent transactions in real-time and can then learn new patterns and rules from each classification which it makes. The goal is not just to develop a credit card fraud system but to also contribute to the existing literature by proposing modifications and improvements for existing systems to make them more accurate and reliable.

**Objectives:**

To achieve the aims that we have highlighted above, there are a set of objectives that we need to accomplish. These objectives are outlined below:

* An analysis of known works in fraud detection, in general, and credit card fraud detection, in specific in literature has been carried out. Evidence of this analysis can be found in the Chapter 2 where we have presented a literature review of the subject matter. We will make use of the methodologies described in these studies with our knowledge to develop the credit card fraud system.
* A credit card fraud detection system will be developed that will connect directly to the databases of credit card companies, banks, and/or other financial institutes. This system will make use of the machine learning model *Artificial Neural Networks (ANN)* for detecting fraudulent transactions which will be implemented through the Keras library.
* For training the ANN model, training data consisting of legitimate and fraud transactions will be provided so that the system can learn how to classify transactions. The dataset which we have chosen is a publicly available dataset found on Kaggle consisting of PaySim mobile money transactions. The learned ANN model that will be used to identify fraudulent transactions and this will be self-learning model which means that it will make use of the predictions that it makes itself to improve itself.
* Each time a transaction takes place, the fraud transaction detector will analyze the transaction against the ANN model to classify it as legitimate (in which case the transaction will hold) or classify it as fraud (in which case the transaction will be blocked, and human intervention will be required).
* A web-based and mobile-based interface will be provided to the fraud transaction detector that will allow users (such as the bank managers) to view all transactions that have been classified by the system. There will be an option provided for filtering out legitimate transactions and fraud transactions. Additionally, it is proposed that an option for unblocking or reversing fraud transactions will also be provided.
* A report that illustrates all aspects of the experiment, including the design of the system, the requirements, the methodology, implementation including code, findings, and conclusions is to be presented. This report will highlight the suggested approach and explain how the results of the experiment can be used to suggest improvements and modifications in existing systems which are used for credit card fraud detection.

In the next chapter, we present a literature review of the subject matter which includes, but is not limited to, fraud detection through meta-learning, machine learning, and credit card fraud detection through neural networks.

|  |  |  |
| --- | --- | --- |
| **Aim** | **Objectives** | **Importance** |
| Literature review of subject matter – analyse existing credit card fraud detection systems. | Study the literature to identify current and proposed solutions for credit card fraud detection. | To understand existing and proposed solutions as a basis for developing a new system. It is important to understand the challenges associated with these solutions and aim to overcome these challenges to contribute a workable solution to the literature. |
| After experimentation, propose improvements and modifications to the proposed solutions to contribute further to the literature. | To maximize the efficiency and effectiveness of the methodologies in literature. |
| Develop a credit card fraud detection system that can identify and prevent fraudulent transactions. | Find a real-world dataset which contains known fraudulent transactions and non-fraudulent transactions. | The dataset will be the basis for training the ANN model which will be used for identifying fraudulent transactions. |
| Train an ANN model using a pre-processed subset of the dataset. | ANN is a comprehensive classifier model that assures high accuracy and reliability when making predictions. |
| Implement a system that makes use of the trained ANN model for classifying transactions that are input to it. This system will be written in Python using the Keras library and will be used to make predictions about fraud transactions. | This will be an improved credit card fraud detection system which will replace traditional current systems that are rule-based expert systems which are non-flexible and have to be continuously updated. |
| Develop a web-based interface for the credit card fraud detection system in C#. | To enable stakeholders to view information about transactions on the internet. |
| Develop a mobile-based interface for the credit card fraud detection system on iOS. | To enable stakeholders to view information about transactions on the go via their smartphones. |
| Provide a feature for filtering out legitimate and fraudulent transactions from the list of all transactions. | To provide a convenient way of monitoring the system’s performance (how many and which transactions did it classify as fraud). |
| Provide a feature to reverse or unblock fraud transactions that have been classified by the system. | To enable human intervention in the case that the system makes an error in classifying transactions. |
| Outline the application of machine learning, in general, and ANNs, in particular, for the purpose of credit card fraud detection. | Write a report that describes the methodology used to implement ANN for credit card fraud detection in detail with an experiment that shows the results. | For contributing to the literature by analysing the performance of machine learning and neural networks in fraud detection. |
| List down the findings and measure, quantitatively, in terms of accuracy and other cost functions, how efficiently and reliably can the developed system detect fraudulent transactions. |
| Outline the benefits of using ANN as the machine learning model rather than Decision Trees or Naïve Bayes for making predictions and classifying transactions. |

*Table 1: Summary of aims and objectives.*

**CHAPTER 2: LITERATURE REVIEW**

A number of authors have contributed to the literature of data science and machine learning for fraud detection, in general, and credit card fraud detection, in particular. The focus of this chapter is to analyze how machine learning and neural networks can be used for detecting fraud. The studies that have been chosen for this literature review propose solutions based on neural networks for the detection of fraudulent transactions among a pool of fraud and real transactions.

**Fraud Detection Using Meta-Learning:**

One of the earliest researches in the domain of credit card fraud was conducted by Stolfo. [3] In the paper by Stolfo, the authors comprehensively discuss the method of meta-learning for learning a model based on classified credit card transaction records and then using this model for predicting whether a transaction is fraudulent or real.

The system proposed by Stolfo is based on two key components – the first is a local fraud detection agent and the second is a meta-learning agent. As Stolfo highlights that standalone fraud detection systems employed by individual financial institutions are not sufficient to eradicate the issue of credit card fraud. Thus, a unified and global approach is required to solve this problem. Therefore, the system proposed by Stolfo consists of a local fraud detection agent which is responsible for detecting and preventing fraud within a single financial institution; and a meta-learning system which collects and integrates the information from individual local agents to provide a secure infrastructure for sharing knowledge about fraud transactions between financial institutions. [3]

The benefit of using such a system is that the knowledge of fraudulent transactions of individual financial institutions can be shared in a global domain without disclosing proprietary data of these institutions. This centralized and secure meta-learning system ensures that a unified solution for detecting and preventing fraudulent credit card transactions can be adopted. [3]

For the experiment, the authors make use of a dataset from the FSTC containing 500,000 records with 20% i.e. 100,000 of these transactions classified as fraud. To prepare the dataset for the experiment, the authors removed redundant fields and sampled from the large dataset to reduce the size of data for quicker learning. Furthermore, the authors divided the dataset into two parts i.e. the training dataset from the earlier months and the test dataset from the later months. For determining the accuracy of the learned model, the authors made use of a cost function that considers both the True and False Positive rates. [3]

The authors made use of BAYES, RIPPER, ID3, and CART for meta-learning and used a 50%/50% distribution of the fraud and non-fraud training dataset for learning the model i.e. the ratio of fraud transactions to non-fraud transactions was set to 1. The results of the experiment concluded that BAYES was the most accurate meta-learning algorithm and that the 50%/50% distribution resulted in the highest True Positive rate and lowest False Positive rate. [3]

Though, for this project, we are not concerned with the use of the BAYES algorithm, we can conclude that the use of a 50%/50% distribution as training data works well for classifying credit card transactions as fraud and non-fraud.

In the paper by Chan, with is co-authored by Stolfo, a similar meta-learning system is adopted for detecting credit card fraud. Chan emphasizes on the importance of non-uniform class and cost distributions for learning in the study. The authors highlight that skewed distributions which have a non-uniform cost per error are common in the real-world and credit card fraud detection is an example of such as distribution. The primary concern with datasets related to credit card fraud is that the number of fraudulent transactions are quite small as compared to the overall number of transactions. [4] Various authors in the past, including Fawcett [5] and Kubat [6] have discussed the degradation effect which skewed class distributions have on the learning. The authors recognize that a natural class distribution which is highly skewed will not yield the most effective results and classifiers.

The dataset which was used by Chan for the experiment contained 500,000 transactions out of which 20% were fraudulent transactions and 80% were legitimate transactions. The author recognizes that in the real-world, the distribution is much more skewed and this 20:80 ratio is a result of the *cleaned* dataset provided by the financial institution. [4]

For learning from the dataset, there were certain steps taken by the author before applying the learning algorithms. The first step was to divide the dataset into two parts: the training dataset consisting of transactions from the first 10 months, and the testing dataset consisting of transactions from the 12th month. The 20:80 skewed distribution is then divided into four subsets, each with a 50:50 distribution. This is done to reduce the skewness and ensure effective results from the learning process. After the division, the learning algorithm is applied on each of the smaller datasets and then the results are combined through meta-learning from the classification behavior. Chan makes use of BAYES, CART, RIPPER, and C4.5 as the learning algorithms for training the model for fraud detection. [4]

There are two primary benefits of dividing the original dataset into smaller datasets, as recognized by the author. Firstly, the smaller datasets have a 50:50 distribution i.e. they are less skewed than the original dataset and thus they yield more effective results. Secondly, the smaller datasets are independent of each other and thus, these can be run in parallel on different processors which can yield significant improvements in speed, particularly when dealing with very large datasets. [4]

The study conducted by Chan provides us with an effective approach for handling very large datasets which are highly skewed. For credit card fraud detection, we can make use of a multi-classifier meta-learning non-uniform cost technique as applied by Chan. [4]

**Fraud Detection using Artificial Neural Networks:**

Serrano [7] has worked with a sample of tax collection data from Federal Patrimony Department (SPU) for proposing a fraud detection system based on neural networks. The authors describe how fraud detection systems can be based on statistical fraud detection schemes such as BAYES and RIPPER. However, these rule-based classifiers are not as adaptive to complex datasets as Artificial Neural Networks (ANNs) and thus the authors choose to use a neural-network based predictor for the fraud detection.

As we highlighted in Chapter 1, Serrano highlights rule-based expert systems as a current approach towards fraud detection. This current approach is based on a set of predefined rules which are gathered and fed into a computer system through human experience. However, the authors highlight that the lack of adaptability is the major drawback of this current approach. The techniques used for committing fraud are continuously evolving and thus the systems that are in place to prevent fraud should also be evolving on their own. This is not the case with expert systems which require frequent updates in onward to stay current and thus, this current approach is not a suggested model for fraud detection. [7]

The authors propose ANNs as a plausible replacement for expert systems for fraud detection. The primary benefits of ANNs highlighted in the study are that these systems can model nonlinear and complex variables with high accuracy, and that these systems can learn adaptively hence it is a generalized solution when trained correctly. [7]

The major challenges associated with ANNs stated by Serrano is that it is very difficult to decide an optimal structure of the network. Since the performance of an ANN varies according to its structure, an adaptive approach in the form of ALN (Adaptive Logic Network) or GANNA (Generalized Adaptive Neural Network Architecture) should be used. Furthermore, other than this, the authors also state that there should be an imbalance between the positive classifier and the other classifiers (such as fraudulent and non-fraudulent in this case) for an ANN to perform well and thus, there needs to be some preprocessing done on the data to achieve this imbalance if it is not present naturally. Lastly, another drawback highlighted by the authors is that since the network in an ANN is the classifier, it needs to be trained with a certain training dataset which can take a significant amount of time. [7]

The dataset which is available to Serrano is that of financial tax transactions provided by the SPU from the years 2005 – 2010. For training the ANN, the authors make use of data from 2005 – 2009 and then use the learned model to make predictions for 2010 to test the accuracy of the ANN model. [7]

The authors have described an ANN as a computational model which is based on the biological neuron cell structure. They focus on Feed-Forward Networks in which the data flows in one direction i.e. from input layer to hidden layer to output layer only. The chosen model by the authors is that of a Multilayer Perception (MLP) in which each neuron of a layer is connected to a neuron of the next layer. The authors make use of a backpropagation algorithm for refining and minimizing the error at the output and use NRMSE (normalized-root-mean-square-error) and COD (cures of dimensionality) measures for transforming the output value to determine if a certain transaction is fraudulent or not. [7]

Based on the results of the experiment conducted by the authors, they concluded that an ANN with two hidden layers, each with 8 neurons, is ideal i.e. produces the minimal error, for predicting fraud with their dataset. [7] In this similar manner, we will have to test and experiment with various configurations of the neural network to determine the ideal configuration for detecting fraud in our dataset.

We briefly analyze a study by Cannady which focuses on the use of neural networks for misuse detection as part of an Intrusion Detection System (IDS). [8] Though the study does not present an experimental model for an ANN, it highlights the advantages and disadvantages of ANNs. The authors state that flexibility, adaptability, and speed of the neural network are its biggest benefits. Since neural networks can perform an analysis with non-linear variables and can learn from new instances of misuse on its own, it is an excellent model to use with misuse detection. As compared to neural networks, Cannady has stated that currently misuse detection makes use of rule-based expert systems which lack flexibility and adaptability as we highlighted earlier and as Serrano highlighted in his study. [7][8]

The authors highlight how the lack of frequent updates to an expert system by an administrator can degrade the system of the entire system. Furthermore, the authors also highlight how if the attack is divided or occurs over an extended time period, then the expert system will fail to detect it since the expert system will be looking if the particular incident has all flags marked red (i.e. it defies all rules or conditions of the expert system). [8]

Lastly, Cannady describe the disadvantages of ANNs for misuse detection, in particular, and fraud detection, in general. The first disadvantage stated is that neural networks have training requirements in which they require a very large dataset to be trained accurately. This means that you need to have access to sensitive data within the target domain (such as attack sequences in misuse detection) and there needs to be a significant time allocated for training the neural network. Furthermore, the training method and configuration chosen are critical and need to be chosen carefully since these directly affect the performance of a neural network. The second disadvantage is the ‘black-box’ nature of neural networks. This means that when a neural network is trained, it is able to achieve a certain level of acceptance but the basis (or function) of this accuracy of the neural network is not known. This is because the neural network automatically adjusts its weights and functions depending on the data fed into it. [8]

**Credit Card Fraud Detection using Artificial Neural Networks:**

Gulati [9] provides a comprehensive analysis on the use of neural networks and geolocation for detecting credit card fraud. The author illustrates that the widespread use of these credit cards makes them a target for hackers and fraudsters which puts credit card transactions at the danger of being tampered with or otherwise altered. The authors define credit card fraud as a type of burglary in which the fraudster obtains cash or makes an installment without the consent of the credit card owner. Gulati further highlights that an effective fraud detection system is one that can: recognize fraudulent transactions in quick time; and distinguish between an imposter and a legitimate customer reliably. The authors also recognize that most existing systems work on credit card transactions after they have been carried out, but the focus of their study is to propose a method that can detect transactions in real-time (i.e. while they are in process). The system that is proposed by Gulati recognizes customer behavior patterns and makes use of this information to mark a transaction as fraudulent if the customer deviates from the regular pattern of making a transaction. The author recognizes that the issue with current (traditional) fraud detection systems is that fraud transactions are detected only after the transaction has been completed after which it is very difficult to track, reverse, and prevent the transaction from taking effect. On the other hand, the author suggests that the use of an ANN-based fraud detection system can help identify fraud transactions while they are being processed (i.e. in real-time) to maximize effectiveness and minimize losses.

Gulati discusses the various kinds of credit card frauds and states that these can be grouped into three classes: card-related frauds, dealer-related frauds, and web-related frauds. Card-related frauds can be either in the form of application frauds where the fraudsters hacks or controls the application being used to acquire sensitive data of the credit card holder; or in the form of stolen cards where the fraudster, through any means, physically takes the card and hence acquires sensitive information of the credit card holder. Dealer-related frauds can be either in the form of merchant collusion in which the dealer themselves intentionally pass on the sensitive data to the fraudsters; or in the form of triangulation in which the fraudster acts as an authentic dealer and presents offers and discounts to convince the card holder to make a transaction to acquire the sensitive data of the credit card holder. Web-related frauds can be false merchant sites which are similar to triangulation but in this, fraudsters make use of phishing sites that simulate known e-commerce websites to capture sensitive data of the credit card holder. The authors further recognize that keystroke logging and mobile phone camera scams are modern techniques which fraudsters use to acquire the information of credit card holders. [9]

The system that is proposed by Gulati classifies transactions as either suspicious transactions which need to go through a verification process to be declared as a valid transaction or a non-suspicious transaction which is directly declared as a valid transaction. Through this methodology, a transaction is marked as suspicious if it deviates from the location pattern or the expenditure pattern of a particular user. For instance, if the user makes a transaction for a very large amount while he normally makes transactions for very small amounts, the transaction will be marked as suspicious. Similar, if the user makes a transaction from a new or unknown location then it will be marked as suspicious. If a transaction is marked as suspicious then the bank and the user is alerted, and the user must go through another round of verification to confirm the transaction. If the system fails to verify a user, then the transaction is declined and marked as fraudulent. In the case the user makes a transaction from a new location and is verified, the new location is automatically added to the location pattern for the user to ensure that future transactions from that location are not marked as fraudulent. For finding the location of the user, the systems make use of an API for public IP lookup. [9] In our study, we will not be making use of geo-location for classifying transactions since we do not have access to the public IP information of users in our dataset.

Gulati [9] makes use of the multilayer perceptron (MLP) model for neural networks which is forward directed. The author recognizes that the hidden layer is one of the most vital parts of the neural network and in their particular case, the lesser the number of neurons, the better the accuracy of the output was. The authors work with sample data from UCI Machine Learning Repository and make use of Java with the Neuroph framework to deploy the ANN. The author concludes that through the use of a fully-developed ANN-based fraud detection system, we can classify transactions with an accuracy of up to 80%. The only drawback that the author identifies with this approach is that new cardholders (who have no transactional history) will not have any input data for the ANN model and thus their transactions will not be classified.

Mishra [10] presents another relevant study on the use of artificial neural networks for identifying credit card fraud. Mishra states that credit card is the most popular method of payment online, but it is subject to fraud through which the fraudster intends to acquire goods without having to pay for it. The financial loss that occurs affects both the financial institutions and individual users. The authors classify credit card fraud as either physical fraud in which the credit card is stolen and used to illicitly purchasing goods; or virtual fraud in which the fraudsters misuse the credit card details of a user online (via the internet). Mishra states that the four methods of credit card fraud include theft fraud, behavioral fraud, application fraud, and bankruptcy fraud.

The authors further highlight that financial institution make use of systems such as credit card authorization, rule-based fraud detection, and address verification systems (AVS) to identify fraud but these are not very effective since fraudsters come up with adaptive techniques to bypass these fraud detection systems. [10]

The ANN architecture used by the authors for their system is a multi-layer feed forward back-propagation network which has 2 hidden layers with 10 neurons and 20 neurons respectively. For computing the accuracy of the model, the authors make use of a simple comparison of the predicted output against the actual output i.e. no cost-sensitive function is used for computing the accuracy of the model. For the activation function, the Sigmoid function is used, and a two-phase model is implemented. In the first phase, the credit card holder’s credentials are verified through user authentication. In the second phase, the neural network is used to predict whether a transaction is valid or not. In case the user cannot be authenticated, *or* the neural network classifies a transaction as not valid, the transaction is rejected and marked as fraudulent in the latter case. [10]

Mishra [10] compares the various techniques that have been adopted for fraud detection systems and puts the focus on testing the accuracy of the BR, GDA, and LM algorithms which are used commonly with ANN models. The data that the author uses consists of two different samples which are obtained from the UCI Machine Learning repository. The authors make use of the Neural Network Train (NNTrain) tool with MATLAB for simulation and breaks down the dataset into three parts: 75% for training the model, 15% for testing the model, and 10% for validating the model. After testing the various techniques of ANN for fraud detection, the author concludes that the BR technique is the most accurate with accuracy scores of 98.745 and 99.01% in the two respective datasets while the GDA technique was the second most accurate and LM was the least accurate. The author also highlights that the accuracy of the BR technique improved as the size of the sample increased. This paper provides a useful insight into how the ANN model should be trained and what algorithm should be used with the ANN to drive the most accurate results for fraud detection.

To summarize, numerous authors have contributed to the literature of fraud detection through neural networks. Most authors recognize that neural networks are a self-growing, adaptive, and modern tool for detecting fraud efficiently and accurately as opposed to rule-based systems that are inflexible and rigid.

Stolfo and Chan presented two of the earliest works on fraud detection and proposed the use of a meta-learning based system. [3][4] Serrano and Cannady presented the drawbacks of rule-based systems and highlighted how neural networks can be effective replacements for such rigid systems. Each of these authors discussed the use of a neural network-based fraud detection system that could be used for applications such as intrusion detection or fraud detection in tax data. [7][8] Gulati and Mishra are the most relevant work in literature for our project since these discuss the use of artificial neural networks for credit card fraud detection. Gulati makes use of a multi-layer perceptron architecture of neural networks with known patterns of behavior for fraud detection and Mishra makes use of a feed-forward back propagation architecture of neural networks for fraud detection. [9][10] We make use of the works of all these authors for proposing an efficient system for credit card fraud detection.

In the next chapter, we discuss the requirements and design considerations for the project which will be used for implementing the credit card fraud detection system.

**Chapter 3: Requirements and Design**

**Requirements:**

The requirements that need to be fulfilled for this project are as follows:

* **Dataset:** A financial dataset needs to be acquired which contains transactional data from a financial institution. It is important that this dataset is either acquired directly from a financial institute such as a bank or it simulates real-world data. This is because the complexity of the variables and features in the dataset will define how the ANN is trained and used. The dataset should contain information about the class i.e. whether a specific transaction is fraudulent and legitimate. Ideally, the dataset should contain more than 1 million records so that the ANN can be trained accurately.
* **Stakeholders:** The stakeholders involved with this project are credit card companies, banks, and other financial institutions. The cooperation of these stakeholders is important since the dataset which will be used to train the ANN model and deploy the system will be sensitive data acquired from them. Furthermore, stakeholders will be able to better advise on what features should be included in the system to minimize the chances of fraudulent transactions. Once the initial experimentation phase is complete, we can pitch the system to these stakeholders with the findings of the trained ANN model to get their support for pushing this system further into implementation phase.
* **Database:** A database that has been built on MySQL will be required for storing the transactional data. This database will be filled from the dataset which is used for training the ANN model. Through a relevant API (Application Programming Interface), this database will then be connected to the system to feed transactional data as input to the learned ANN model. Once the ANN model makes a prediction, the information will then be stored back in the database.

**Description of the Dataset:**

The dataset which we have chosen for this report consists of mobile money transactions which have been simulated from real transactions. The data has been contributed by E.A. Lopez-Rojas and it has been extracted from the logs of a mobile money service provider based in an African country. The PaySim mobile money service provider is a multinational company and it has its services running in 14 countries across the world. The dataset that we have used is a scaled down version which is available on Kaggle and it consists of 5,090,096 mobile money transactions. Out of these transactions, only 0.13% are classified as fraudulent transactions. Therefore, the dataset has a skewed distribution and we need to apply appropriate measures to refine and reorder the dataset. This needs to be done in order to ensure high accuracy and effectiveness of the learned model.

**Why Neural Networks?**

A question that arises here is: Amongst all the machine learning algorithms which include Decision Tree classifier and BAYES classifier, why are Neural Networks preferred for this dataset? For complex and large datasets in which there is no well-defined relationship between the various variables (i.e. attributes) of the data, Decision Tree classifier and BAYES classifier are not known to yield accurate results. Not just this, but it is increasingly difficult to structure and apply other classifiers to a problem such as this where we are looking to recognize a pattern and make use of this pattern for identifying and preventing fraudulent transactions from occurring in the future. In a real-life problem such as this where the relationships are non-linear and the dataset is skewed to favor a particular class, we can make use of the neural network to apply the necessary mathematical transformations such as the ReLU and Sigmoid function to learn an accurate model from the dataset. The accuracy and efficiency of neural networks in solving complex problems such as this is the primary reason why we have made use of this machine learning algorithm for our report.

**Structure of the Neural Network:**

In Chapter 1, we outlined the structure of a neural network which is used for learning models and classifiers for making predictions. Here we outline how we will make use of that structure to build our neural network model. For our experiment, the neural network will consist of 3 layers i.e. a simple perceptron. The first layer i.e. the input layer will consist of 9 neurons each of which will be mapped from 9 columns (i.e. features) found in the dataset. For the second layer i.e. the hidden layer, we will be testing a variable number of neurons to determine the ideal configuration for optimum accuracy. Ultimately, we will be making use of up to 4 hidden layers in the neural network for learning the model from the dataset. The final layer i.e. the output layer will consist of a single neuron which will be used to determine whether a specific transaction is fraudulent or legitimate.

***<<insert structure of neural network here>>***

**Activation Function: Rectified Linear Unit and Sigmoid:**

For the activation function which will be used for transforming the inputs at neurons, we will be making use of Rectified Linear Unit (ReLU) and Sigmoid. We have described both of these functions briefly below.

Rectified Linear Unit (ReLU) is a ramp function which was introduced and demonstrated in 2011 due to its advantages over traditional activation functions such as logistic sigmoid. Unlike other activation functions, ReLU does not saturate since the gradient is always either 1 or 0 and it adds non-linearity to the network. ReLU learns faster than traditional activation functions but the drawback is that it does not activate for negative inputs and thus, all negative inputs are regarded as dead neurons with ReLU.

Sigmoid is a traditional activation function which is very commonly used for various kinds of neural networks. Though it does not learn as fast as ReLU, Sigmoid is an effective activation function that can activate both positive and negative inputs.

**Adam Optimizer:**

For the choice of optimizer, we are making use of Adam. Adam is a stochastic optimization algorithm which makes use of first-order gradient-based optimization. The reason for choosing the Adam optimizer is because it has low memory requirements, is computationally efficient, and generally well suited for datasets that are large in size or has a high number of parameters. The Adam optimizer inherits properties from both the RMSProp optimizer and AdaGrad optimizer which are two of the best optimizers present.

It is important to mention here that the Keras library which we will be using for this experiment has built-in functionality for the activation function of ReLU and Sigmoid and the Adam Optimizer.

**Data Refinement:**

For reducing the learning time and improving the accuracy of the learned model, we will be removing redundant, duplicated, incomplete, and otherwise erroneous transactions from the dataset. In addition to this, since the size of the dataset is quite large i.e. 5,000,000+ transactions, we will be randomly sampling transactions from the dataset without replacement.

**Multi-Class Meta-Learning:**

As with the real-world, the dataset which we are making use of for this experiment has a skewed distribution with only 0.13% of the transactions marked as fraud. Due to the dominance of legitimate transactions in the dataset, there is a high probability of false negatives i.e. fraudulent transactions being marked as legitimate transactions by the model. To diminish the effect of this skewness, we will make use of the multi-class meta-learning model which was proposed by the authors Stolfo and Chan as we discussed in Chapter 2. [3][4]

Ideally, we will be looking to divide the dataset into smaller datasets that have a 70:30 distribution i.e. 70% fraudulent transactions and 30% legitimate transactions. The reason for dividing the data as such is because our objective is to detect fraudulent transactions and thus, minimize the False Acceptance Ratio (FAR). A low FAR means that there are very few fraudulent transactions that are being marked as legitimate. An undetected fraudulent transaction can have a significant cost and therefore we need to minimize the probability of this happening. For this experiment, we can trade-off a high False Rejection Ratio (FRR) for a False Acceptance Ratio (FAR) since the cost of a fraudulent transaction being marked as a legitimate transaction is much higher than the cost of a legitimate transaction being marked as a fraudulent transaction.

**Accuracy Considerations and Cost Function:**

Credit card fraud is a sensitive matter and a single set of fraudulent transactions can lead to loss worth millions of dollars for the bank, accountholders, and other stakeholders. For this reason, we cannot make use of a simple ratio between correct predictions over total predictions for the accuracy of the learned model. Instead, we have to evaluate the cost which is associated with a False Negative i.e. when a transaction is fraudulent, but it is marked as legitimate and with a False Positive i.e. when a transaction is legitimate, but it is marked as fraudulent. We will make use of a cost matrix for this purpose where we will evaluate the Precision, Recall, and F-Measure for the learned model to determine how accurate, consistent, and effective the predicted results are.

The next chapter describes the proposed methodology for implementing the credit card fraud detection system which we have discussed.

**Chapter 4: Methodology**

In this section, we discuss the process that we will use to implement the system. The following steps are required for designing, implementing, and training the artificial neural network-based credit card fraud detection system.

1. **Acquire the dataset:**

The dataset is an integral part of this project since it is what will be used to train, test, and validate the system. We have outlined the essentials of the PaySim mobile money dataset which will be using for this project in the previous section. The original dataset has been used and can be found in the paper *“PaySim: A financial mobile money simulator for fraud detection”.* However, for this project, we will be making use of a scaled-down version (1/4th of the original dataset) which can be found on Kaggle. The dataset has been made available on [Kaggle](https://www.kaggle.com/ntnu-testimon/paysim1) by the original owner because there is a lack of publicly available financial services dataset. The dataset contains 5,090,096 transactions in total. There are a total of 11 columns in the dataset which include step, type, amount, nameOrig, oldbalanceOrig, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest, isFraud, and isFlaggedFraud. Though the columns are self-explanatory, the description of each of these columns can be found at the source of the dataset. Out of these 11 columns, we will be making use of 9 columns as the input to our system since these are the variables that determine the value of the last 2 columns i.e. the output from our system. It is worth mentioning here that not all of these columns (or variables) are numerical – some of these contain string values. Since neural networks only allow for numerical inputs, we will have to transform these string values into numeric values on an appropriate scale before training the neural network.

1. **Create a model for the neural network:**

We will make use of a multi-layer perceptron architecture for our neural network which will pass information in a forward direction with back propagation for weight adjustment and error correction. In our implementation, the neural network will have the following configuration:

*Type of network: Feed forward back propagation multi-layer perceptron*

*Number of neurons in input layer: 9*

*Number of hidden layers: 4*

*Number of neurons in hidden layer # 1: 14*

*Number of neurons in hidden layer # 2: 28*

*Number of neurons in hidden layer # 3: 14*

*Number of neurons in hidden layer # 4: 7*

*Number of neurons in output layer: 1*

*Training algorithm: Adam Optimizer*

*Maximum number of iterations: N*

*<<insert neural network structure here>>*

1. **Configure the neural network:**

The neural network will be configured as per the model that we have outlined in the previous step. For configuring the neural network, we will make use of the Keras framework based on Python in which we will define the input layer, hidden layers, and output layer. Furthermore, we will define the activation function as Rectified Linear Unit (ReLU) and Sigmoid function as per the equations given below.

***Rectified Linear Unit (ReLU):*** *f(x)=max (0, x)*

***Sigmoid:*** *σ(x)=1/(1+e−x)*

Keras has these activation functions built-in and thus, we will not have to implement these formulae manually. However, it is important to have an understanding of these formulae to know how they affect the performance of a neural network. For instance, ReLU is thresholder at zero and thus does not allow for negative inputs to it. We have discussed both activation functions in detail in the previous section.

As for the Adam Optimizer, it requires the following parameters:

* α — The learning rate or the step-size. It has an optimal value of 0.001.
* β1 — The exponential decay rate for the running average of gradient (first moment). It has an optimal value of 0.9.
* β2 — The exponential decay rate for the running average of square of the gradient (second moment). It has an optimal value of 0.999.
* ε – Used to prevent the error of division by zero. It has an optimal value of 10-8.

Given the following parameters, the Adam Optimizer algorithm can be used to determine and adjust the error in the weights of the neural network. Keras also features a built-in implementation for the Adam Optimizer as well and therefore, we do not need to implement the algorithm from scratch. However, we do need to provide it with the required parameters and use the results of the algorithm to update the weights of the neural network as we will demonstrate in the next section.

1. **Initialize weights of the neural network:**

A neural network must be initialized with weights before the first iteration of the training to provide sufficient parameters for the algorithm to work. Without the weights, the forward transfer of information would not be possible. However, it is important to choose the initial weights carefully. The initial weights of the neural network influence the accuracy and pace with which the neural network is trained. If the weights are not set correctly, it can take very long to train the neural network. There are two possible ways to initialize the weights of the neural network: the first is to initialize them randomly, and the second is to initialize them through some known heuristic or known value. However, in our case, since no heuristic is known, we will initialize the weights of the neural network randomly. A weight will be assigned to each edge in the network model.

1. **Train the neural network using the training dataset:**

For training the neural network, we divide the dataset into two parts – training and testing. This is done to ensure that the neural network can be tested against known outputs to determine how accurate it is – if the entire dataset is used to train the neural network then we would not have known outputs to determine whether a prediction is correct or not. We will divide the dataset into two sets – the first set will consist of 80% of the data and this will be the training dataset; the second set will consist of 20% of the data and this will be the testing dataset. The training dataset will then be further divided into smaller subsets to have a high ratio for the number of fraudulent transactions to the number of legitimate transactions. This will ensure that the False Acceptance Ratio (FAR) can be minimized. Ideally, we will look to have a ratio of 70:30 for fraudulent transactions to legitimate transactions. However, this is not possible with real-world transactions since the number of fraudulent transactions are quite low when compared to legitimate transactions in such a scenario. We will make use of random sampling for dividing the dataset into training and testing datasets to ensure that there is no biasness.

Once we have the training dataset ready, the next step will be to make use of this dataset for training the neural network model. As we have already discussed, there will be 9 inputs to the neural network each of which represents a single column in the original dataset. Each transaction in the training dataset will then be input to the neural network which will then pass through to the hidden layers and the output layer after processing. Once the output value is obtained, the Adam Optimizer algorithm will be used to compute and adjust the error in the weights of the neural network. The same process will be repeated for each transaction in the training dataset. After all transactions have been input to the neural network, 1 iteration of the training phase will be complete. The entire process will be repeated until the accuracy of the neural network can no longer be improved further. Once N iterations are complete, in our case, we will analyze the performance of the neural network. After this, we will retrain the neural network to achieve a desired higher accuracy. In the case of neural networks, it is known that the higher the number of retrains, the higher the accuracy of the model trained.

1. **Test the neural network using the training dataset:**

After the neural network has been trained, we will test how accurate and reliable it is through the testing dataset. This step is carried out to ensure that the neural network meets the required threshold for accuracy. If the accuracy is not up to the required threshold then the configuration of the neural network is adjusted, and the neural network is retrained. The adjustments that are made to the configuration of the neural network can be to increase or reduce the number of hidden layers, increase or reduce the number of neurons per hidden layer, and alter the weights of the neural network. This process is repeated until the desired accuracy is achieved.

It is worth mentioning here that when we pass the inputs through the neural network to compute the output, no additional step will be taken after this to adjust the weights of the network i.e. the Adam Optimizer will not be run. This is because the neural network is now trained for this step and thus we are only concerned with the output value and not the configuration of the neural network itself.

For determining the accuracy of the trained neural network model, we will first pass all inputs in the testing dataset through the neural network to obtain the predictions (outputs) on which transactions are fraudulent and which are legitimate. Each output, which we will call the predicted output, will then be recorded in a table with the actual output (whether the transaction was fraudulent or legitimate in the actual testing dataset). Once we have recorded all the predicted outputs against the actual outputs, we will compute the accuracy, precision, recall, and F-measure for the neural network. The accuracy is simply the ratio of the number of correct predictions to the number of total predictions. The precision is the ratio of the number of correct positive (fraudulent) predictions to the number of total predictions. The recall is the ratio of the number of correct positive predictions to the number of total actual positive transactions. Both precision and recall are cost-sensitive measures that are concerned with how relevant (i.e. how many fraudulent transactions) were correctly predicted. The F-measure is a function of precision and recall that gives us a good estimate of whether the neural network is accurate and unbiased or not. The F-measure is given by the formula: 2 \* (precision \* recall) / (precision + recall).

1. **Demonstrate the results through simulations and graphs:**

Once the neural network for the credit-card fraud detection system is ready, we will make use of Python-based simulations to demonstrate the results. These results will show how various parameters of the neural networks such as different activation functions and different number of neurons per hidden layer affect the performance of the neural network. These simulations will be used in Section 6 of this report to demonstrate the findings of the project and draw a conclusion for the effectiveness of a neural network based fraud detection system.

1. **Make use of the neural network for classifying future transactions:**

This is the next step forward with the project which we will not be implementing in the scope of this report. After the neural network has been trained and tested, the next step will be to connect the neural network with the database of the financial institute. After this, the neural network will automatically classify all transactions that are made online and the results will be displayed on a web-based interface to the stakeholders.

In the next chapter, we will make use of the methodology which we have outlined in this chapter to implement the neural network based credit card fraud detection system. All implementation details, including the tools used and exact parameters defined will be described in detail in the next chapter.

**Chapter 5: Implementation**

In this chapter, we discuss details regarding how the proposed fraud detection system has been implemented. As we mentioned in the first chapter, the credit-card fraud detection system makes use of an artificial neural network implemented on Python and Keras library for making predictions regarding incoming transactions. The development environment which has been used for implementing the system is Spyder 3.6 with the open source Anaconda distribution for Python. Git has been used as the version control for the software via GitHub.

**Input Transformations:**

The one-hot encoding technique has been used for transforming the input in our PaySim mobile payments dataset. As mentioned in the previous chapter, neural networks can only take numeric variables as input and therefore, there is a need to transform all String variables to numeric variables through an appropriate method, which is one-hot encoding in our case.

Through the one-hot encoding technique, categorical variables (i.e. variables that contain String values) are represented as binary vectors. For this, each unique value of the variable is assigned an integer value (which is generally in increasing order starting from 0). After this, a binary vector will be generated that will have a length equal to the number of unique values for that categorical variable. This is how one-hot encoding is carried out and it is preferred over simple integer encoding or label encoding because it is more expressive and it works for variables that have no ordinal relationships i.e. a higher value does not mean that it is a better category.

In our implementation, we have made use of a built-in function, ***to\_categorical()***, in the Keras library to transform a categorical variable into a binary vector. This transformation enables the dataset to be input to the neural network. Once all the processing has been done on a transaction and it is output by the transformation, then we can make use of the **decode()** function in the Keras library to transform the binary vector back into a categorical variable.

**Sample Inputs and Outputs:**

Here is a sample of transactions in the dataset which have been transformed for input to the neural network via the one-hot encoding technique.

*<<insert sample inputs table here>>*

Here is a sample of the output that is generated by the neural network which will be used to determine whether particular transactions are fraudulent or legitimate.

*<<insert sample outputs table here>>*

**Neural Network Implementation:**

The neural network for the system has been implemented using the Keras library for Python. The first step that we take is to load the dataset into the system via the Numpy library in Python. After this, we make use of the method outlined in the section above to transform the dataset into the desired format as input to the neural network.

The next step is to define the model for the neural network. For this, we make use of the **Sequential** class in the Keras library to create the model and then the **add()** function to add each layer of the neural network one by one. The first layer is the input layer with 9 neurons, followed by 4 hidden layers with 14 neurons, 28 neurons, 14 neurons, and 7 neurons in each respectively, followed by the output layer which has just 1 neuron. In this step, we also define the activation functions for the neural network which are **relu** and **sigmoid** in our case.

After the model has been defined, we make use of the **compile()** function to generate the neural network which will be used for fraud detection. In this step, we set the optimizer for the neural network to be the **Adam optimizer.**

Once the model for the neural network is ready, we train (or fit) it with our training dataset. For this, we make use of the **fit()** method in the Keras library and provide it with the parameters for the training dataset. We define the number of iterations N and the number of transactions after which the weights of the neural network are updated in this step as well. After this step, we have successfully developed our model for the neural network and can now make predictions about transactions using it.

The next step is to evaluate the model or test the accuracy of the neural network that we have prepared for the fraud detection system. For this, we make use of the **predict()** method in the Keras library and provide it with the testing dataset as the parameter. We then apply the appropriate formulae on the output results to determine the accuracy, precision, recall, and F-measure of the predictions to determine how consistent and reliable the neural network is.

The steps that we have outlined in this section were repeated a number of times in order to reach to the optimal configuration which we have outlined and used for the system. To maximize the accuracy of the model, the number of iterations, number of hidden layers, number of neurons per hidden layer, and the ratio of fraudulent transactions to legitimate transactions in the training dataset had to be adjusted a number of times.

After the neural network has been developed, we generate simulations including graphs and charts to demonstrate the results of the fraud detection system and our findings for this report.

**Simulations:**

Simulations for demonstrating the results output from the neural network have been generated through the use of an independent Matplotib library for Python. Matplotlib is a graph plotting library for Python that serves as an extension to the numerical mathematics extension, the NumPy library.

It is important to mention here that while simulations are not part of the fraud detection system itself and are not outlined in the initial requirements for the system, these are important because they demonstrate our findings in a visual and meaningful manner. These simulations have been generated to assist in comparing the performance of this neural network-based credit card fraud detection system with that of other systems proposed in literature and traditional systems.

For generating the simulations, we make use of the output values that are generated by the neural network with its measures of accuracy and configuration to plot graphs comparing the performance of the various configurations used for the neural network. We generate a variety of graphs to demonstrate the results via the Matplotlib library which include sinusoidal graphs.

**Interface for the System:**

For the end-user and the stakeholders of the fraud detection system, the complexities of the neural network and how predictions are made are not important. Instead, they need an interface where they can identify fraudulent transactions and take relevant actions for managing these transactions such as reversing or accepting the transaction. For this purpose, a web-based interface has been developed using the programming language C# and a mobile-based interface has been developed for iOS using the programming language Swift.

The features that are part of the implemented interface are as follows:

* Add a set of transactions that need to be predicted as either fraudulent or legitimate using the .csv file format.
* View all transactions that have been classified and sort the results using the type of transaction (i.e. filter out fraudulent transactions and legitimate transactions separately).
* On each transaction, carry out tasks such as reversing the transaction, deleting the transaction, or accepting the transaction.

At the back end for this interface, the Python-based neural network will be connected which will be used for classifying the transactions as either fraudulent or legitimate.

The web-based interface which has been developed for the fraud detection system is shown in the figures below.

*<<insert web application figures here>>*

The mobile-based interface which has been developed for the fraud detection system is shown in the figures below.

*<<insert mobile application figures here>>*

In the next chapter for this report, we will outline all of our findings from this research and project. We will compare the performance of the various configurations of the neural network for fraud detection system and highlight how different configurations will be suitable for different kinds of datasets. We will also compare the performance of our neural network based fraud detection system with traditional systems to illustrate how neural networks are the next step forward for improving fraud detection and prevention in the future.

**Chapter 8: References**

[1] Tatham, M. [2018]. *Identity Theft Statistics*. [Online] Available from: <<https://www.experian.com/blogs/ask-experian/identity-theft-statistics/>> [Accessed: 12 July 2018]

[2] Steele, J. [2017]. *Credit card fraud and ID theft statistics*. [Online] Available from: <<https://www.creditcards.com/credit-card-news/credit-card-security-id-theft-fraud-statistics-1276.php>> [Accessed: 12 July 2018].

[3] Stolfo, J., Chan, K. [1997]. *Credit Card Fraud Detection Using Meta-Learning Issues and Initial Results.* [AAAI] Available from: <<http://www.aaai.org/Papers/Workshops/1997/WS-97-07/WS97-07-015.pdf>> [Accessed: 19 April 2018].

[4] Chan, K., Stolfo. J. *Towards Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection.* [AAAI] Available from: <<http://www.aaai.org/Papers/KDD/1998/KDD98-026.pdf>> [Accessed: 19 April 2018].

[5] Fawcett, T. 1996. *Learning with skewed class distributions summary of responses*. In Machine Learning List, Vol. 8, No. 20.

[6] Kubat, M., and Matwin, S. 1997. *Addressing the curse of imbalanaced training sets: One sided selection*. In Proc. l4th Intl. Conf.Machine Learning, 179-186.

[7] Serrano, A. M. R., Cardonha, C., Fernandes, A., de Sousa, R. T. [2012]. *Neural Network Predictor for Fraud Detection: A Study Case for the Federal Patrimony Department.* DOI: 10.5769/C2012010

[8] Cannady, J. *Artificial Neural Networks for Misuse Detection.* Available from: <<http://pld.cs.luc.edu/courses/intrusion/fall05/cannady.artificial_neural_networks_for_misuse_detection.pdf>> [Accessed: 18 July 2018]

[9] Gulati, A., Dubey, P., MdFuzail, C., Norman, J., Mangayarkarasi, R. [2017] *Credit card fraud detection using neural network and geolocation.* [IOPScience]. DOI: 10.1088/1757-899X/263/4/042039

[10] Mishra, C., Gupta, D. L., Singh, R. [2017]. *Credit Card Fraud Identification Using Artificial Neural Networks. [IJCS]. Available from:* <<https://pdfs.semanticscholar.org/1eaa/5af431d96daf89ee14134aaa628c4dee814b.pdf>> [Accessed: 22 July 2018].