Fraudulent Transaction Detection and Prevention Using Deep Neural Networks

by

MOHAMMED NAJIM UDDIN

Thesis

Submitted in fulfilment of the requirements for the degree of

Master of Research

Central Queensland University

School of Engineering and Technology

Names of Supervisory Panel:

Dr Rahat Hossain, Dr Salahuddin Azad and Associate Professor Ritesh Chugh

February 2024

Abstract

The worldwide explosion of information technology has significantly streamlined financial transactions via electronic devices like computers, mobile phones, and tablets, all made possible by the ubiquity of the Internet. While this has provided significant convenience in settling financial transactions, it has also led to a severe threat, as evidenced by the increasing incidents of fraudulent transactions. These fraudulent activities result in financial losses for both financial institutions and customers. Card fraud incidents increased by 14.34% to $566 million in 2022, compared to $495 million in 2021 in Australia. To combat these fraudulent transactions, researchers have been actively exploring various methods, including but not limited to statistical, machine learning, and deep learning techniques. Given the effectiveness of deep learning in various artificial intelligence applications, this has demonstrated better performance than traditional machine learning in detecting fraudulent transactions.

A multi-pronged approach was adopted for the conduct of this study. This project explores deep learning techniques for identifying and preventing fraudulent transactions. Deep learning is a machine learning approach utilising artificial neural networks with multiple layers to automatically extract hierarchical representations from data, enabling complex pattern recognition. This research involves generating features from financial transactions to train the deep learning models. Additionally, it emphasises the optimisation of the deep learning model and proposes an efficient approach for classifying fraudulent transactions. After reviewing the existing literature, it was found that fraud detection faces several challenges. Firstly, the data is imbalanced, meaning fraudulent transactions are considerably less frequent than non-fraudulent transactions. Secondly, fraudulent data is scarce due to organisations' reluctance to disclose such data for privacy and reputational concerns. We have also explored several classical and deep machine learning to detect fraudulent transactions, such as Logistic Regression (LR), Conventional Neural Networks, K-Nearest Neighbour (KNN), Decision Trees, Random Forests, Convolutional Neural Networks (CNN), Autoencoders, Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM), and Deep Belief Network (DBN), among others. Although these methods perform well in detecting fraud, there is room for improvement by employing techniques such as hyperparameter tuning, feature engineering, and mitigating overfitting issues.

Two novel methods were thus proposed for improving the efficiency of deep learning models for fraud detection. Firstly, a feature engineering technique called Deep Feature Synthesis (DFS) was deployed with the CNN model to develop a new model. This feature engineering technique automatically generates features from existing features by applying mathematical functions based on the relationship among the datasets. Following this, an optimised CNN model was developed by fine-tuning the hyperparameters. Two algorithms, random search and grid search were employed for hyperparameters optimisation. A step-by-step process was followed in tuning the hyperparameters, where each hyperparameter was adjusted within the defined search space, tested with the test data, and subsequently updated the model with the tuned hyperparameters. This process continued until the optimised hyperparameters were identified. A research methodology was designed to validate and test the efficacy of the proposed improvements, including data collection, pre-processing, data splitting, model training, optimisation, and model testing and evaluation. An analysis of DFS's impact on CNN's performance regarding the accuracy, precision, recall, F1 score, and AUC was conducted. The results demonstrated a significant improvement, with performance increasing by 11% in all four indices after incorporating DFS in the pre-processing step of the CNN deep learning model training compared to the standalone CNN model. On the other hand, the CNN model optimised through random search showed a performance enhancement of 7% over the baseline and a 10% improvement compared to the existing model in the current literature, and the optimised CNN using the grid search Algorithm exhibited a 5.5% enhancement over the baseline and an 8.9% improvement over the existing model.

Overall, this study focuses on detecting fraudulent transactions using deep learning techniques, specifically Convolutional Neural Network (CNN) models. A systematic literature review is conducted to address research questions, exploring six experimental steps for fraudulent transaction detection using deep learning. The integration of DFS improves fraud detection performance, resulting in a notable enhancement in recall, F1 score, and accuracy. Hyperparameter tuning, utilising grid search and random search algorithms yield remarkable improvements across precision, recall, and F1 score for the optimised CNN models compared to the baseline. These optimised CNN models surpass the baseline and outperform a state-of-the-art method.

CANDIDATE'S STATEMENT

By submitting this thesis for formal examination at CQUniversity Australia, I declare that it meets all requirements as outlined in the Research Higher Degree Theses Policy and Procedure.

STATEMENT AUTHORSHIP AND ORIGINALITY

By submitting this thesis for formal examination at CQUniversity Australia, I declare that all of the research and discussion presented in this thesis is original work performed by the author. No content of this thesis has been submitted or considered either in whole or in part, at any tertiary institute or university for a degree or any other category of award. I also declare that any material presented in this thesis performed by another person or institute has been referenced and listed in the reference section.

COPYRIGHT STATEMENT

By submitting this thesis for formal examination at CQUniversity Australia, I acknowledge that the thesis may be freely copied and distributed for private use and study; however, no part of this thesis or the information contained therein may be included in or referred to in any publication without prior written permission of the author and/or any reference fully acknowledged.

ATTRIBUTION STATEMENT

Except where indicated, images, maps and/or drawings presented within this thesis are owned by the Author.

PREVIOUS SUBMISSION STATEMENT

This paper has not been submitted for an award by another research degree candidate, either at CQUniversity or elsewhere.

THIRD PARTY COPYRIGHT STATEMENT

I confirm that my published thesis does not infringe the copyright of any person, persons or organization, or include any material (journal articles, maps, images, figures, music scores etc.) belonging to another party.

ACKNOWLEDGEMENT OF PROFESSIONAL SERVICES

Professional editor, Dr. Jeffrey Keddie, provided copyediting and proof-reading services, according to the guidelines laid out in the University-endorsed Australian national guidelines, 'The editing of research theses by professional editors'.

DECLARATION OF CO-AUTHORSHIP AND CO-CONTRIBUTION

1. **Title of Paper:** Impact of Deep Feature Synthesis on Deep Learning in Electronic Transaction Fraud Detection

Full bibliographic reference: Uddin, M. N., Azad, S., Hossain, M. R., & Chugh, R. (2023). Impact of Deep Feature Synthesis on Deep Learning in Electronic Transaction Fraud Detection. *2023 IEEE 3rd International Conference on Software Engineering and Artificial Intelligence (SEAI)*, 204-208

Status: Published

NATURE OF CANDIDATE'S CONTRIBUTION, INCLUDING PERCENTAGE OF TOTAL

In conducting the study, I was responsible for collecting the data, conducting experiment, analysing result and drafting the paper. This publication was written by me, and my contribution was 70 %.

NATURE OF CO-AUTHORS' CONTRIBUTIONS, INCLUDING PERCENTAGE OF TOTAL

My co-authors, M. R. Hossain (10% of the work), S. Azad (10% of the work), and R. Chugh (10% of the work) contributed to the paper. M. R. Hossain contributed by conceptualising the paper and supervising the study, R. Chugh contributed by reviewing, editing the paper and S. Azad contributed by reviewing and editing the technical part of the paper.

2. **Title of Paper:** Fraudulent Transaction Detection and Prevention using Deep Learning: A Systematic Literature Review

Full bibliographic reference: Uddin, M. N., Hossain, M. R., Chugh, R., & Azad, S. "Fraudulent Transaction Detection and Prevention using Deep Learning: A Systematic Literature Review" Progress in Artificial Intelligence

Status: Under review.

NATURE OF CANDIDATE'S CONTRIBUTION, INCLUDING PERCENTAGE OF TOTAL

In conducting the study, I was responsible for forming the research question, collecting the literature, analysing data and drafting the paper. This publication was written by me, and my contribution was 70%.

NATURE OF CO-AUTHORS' CONTRIBUTIONS, INCLUDING PERCENTAGE OF TOTAL

My co-authors, M. R. Hossain (10% of the work), R. Chugh (10% of the work), and S. Azad (10% of the work) contributed to the paper. M. R. Hossain contributed by developing a search strategy and supervising, R. Chugh contributed by reviewing and editing the paper and S. Azad contributed by conceptualisation and reviewing the technical part of the paper.

Mohammed Najim Uddin

Acknowledgements

I would like to express my sincere gratitude and respect to my principal supervisor, Dr Rahat Hossain, for his continuous guidance, advice, encouragement, support, and every possible help throughout the work and preparation of this thesis. I am also very grateful to my associate supervisors, Dr Salahuddin Azad, and Associate Professor Ritesh Chugh, for their suggestions, time, devotion, and effort guiding me throughout the experiment and thesis writing.

I want to express my gratitude to CQUniversity for providing me with an excellent education service. I also thank Onnorokom Solutions Ltd for partnering with CQUniversity to provide financial support for my study.

I would also like to thank the technical and administrative personnel at CQUniversity for their support throughout my studies. I am especially grateful to all the School of Graduate Research staff for their support and assistance during my study.

I am deeply grateful to my parents, family members, and well-wishers for their support and encouragement. I particularly appreciate my wife, son, and daughters' sacrifices, patience in enduring my absence, and continuous emotional support from a distance.

Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Background

Most global economic transactions are being done electronically. Due to the abundance of handheld devices, laptops, computers, and point-of-sale machines, cashless transactions have gained immense popularity in the last decade. In today's world, digital devices have become indispensable for day-to-day financial transactions, revolutionising how we handle our finances. This convenience simplifies our lives and conserves precious time and energy. Gone are the days of enduring long queues at banks and other financial institutions for simple transactions; now, we can effortlessly manage our finances with ease and efficiency. The reliance on these devices has ushered in a new era of seamless and hassle-free financial interactions, allowing us to focus on more important aspects of life. In addition, technological advancement allows us to execute financial transactions around the clock and purchase goods or services from any corner of the globe. However, these systems of financial transactions systems are not fully secured, and fraudsters are continuously trying to violate security and steal money and personal information. Fraudulent activities cause financial losses to financial institutions and customers. Fraudulent activities also have increased recently. A report published by the Australian Bureau of Statistics shows increased fraudulent incidents from 6.9% in 2020-21 to 8.1% in 2021-22 (ABS, 2022). According to the Payment Fraud Report by AusPayNet (2023) card fraud incidents increased by 14.34% to $566 million in 2022, compared to $495 million in 2021. The report also reveals the total fraud amount was $576 million, $465 million, and $469 million in 2018, 2019, and 2020, respectively.

In electronic financial transactions, two parties exchange sensitive information to complete the transaction. Credit card transactions, fund transfers on the wire, and online shopping are included in the financial activities (Stojanović et al., 2021). Financial activities are vulnerable because they depend on information technology infrastructure. Fraudsters target particular weak points of the system to exploit. Scammers illegally collect identifiable information such as user ID, password, fingerprint, date of birth, address, and system log data to commit identity theft and make online transactions as original users (Ali, 2012). Fraudsters keep trying to discover the weaknesses of the existing fraud detection systems and change their attacking behaviour, which makes it

challenging to identify fraudulent transactions in real-time, which even human experts cannot accurately predict. In addition, credit cards or plastic cards are extensively employed for various financial transactions. Still, unfortunately, perpetrators engage in fraudulent activities by unlawfully obtaining money from the original card owners through fraudulent transactions. Fraudulent transactions of credit cards are mainly categorised into two broad types: application and behavioural fraud (Jha & Westland, 2013). Application fraud is conducted in two ways. The first way is when credit cards are collected using someone else's identity information and are repeatedly used until the cardholder stops it. The second way is when the card is issued by fraudsters using false identity information, and they use the full limit of the card without ever making payments.

Behavioural fraud pertains to monitoring customers' spending behaviour. Behavioural fraud is divided into four subcategories: counterfeit, mail intercept, stolen/lost, and mail and telephone order fraud (Jha & Westland, 2013). Counterfeit fraud is conducted by stealing the information of a genuine customer and creating a physical counterfeit card, which is then used to commit fraud. Mail intercept is a type of fraud where the card is stolen while it is being delivered to the customer after issuance. A different kind of fraud is stolen or lost card fraud, where fraudsters obtain the card through theft. Finally, mail or telephone order fraud is a type of fraud that does not require the physical presence of the card or cardholder. In this type of fraud, perpetrators commit fraud virtually over the phone or online without physically signing or swiping the card. Due to the striking resemblance between fraudulent and non-fraudulent transactions, detecting fraudulent transactions has become a challenging yet vital research topic (Zareapoor & Shamsolmoali, 2015).

It is important to prevent and identify fraud through effective detection mechanisms. In the ever-evolving world of financial technology, transactions occur within milliseconds. However, this progress has also provided new opportunities for fraudsters to exploit vulnerabilities. Cybercriminals continuously develop sophisticated techniques, such as phishing, identity theft, and data breaches, to gain unauthorised access to sensitive financial information and carry out fraudulent activities that threaten organisations' stability and ethical reputation (Kraemer-Mbula et al., 2013). Fraudulent activities can cause significant financial losses, reputation damage, and customer trust

erosion (Smith, 2004). Furthermore, in addition to the financial losses incurred, fraudulent incidents can lead to legal disputes, regulatory audits, and penalties for non-compliance. Detecting and preventing fraud maximises organisational revenue and strengthens the organisation by uplifting customer faith and trust.

Automated fraud detection offers a rapid and efficient solution for preventing and identifying online fraudulent activities. There are several key reasons why automated fraud detection is highly advantageous. Firstly, electronic transactions occur within milliseconds, making it impractical to rely on manual mechanisms for fraud detection. Manual processes not only consume valuable time but also incur significant costs. Secondly, the sheer volume of daily financial transactions presents a formidable challenge. Analysing large volumes of data manually can be tedious and labour-intensive. Automated fraud detection systems excel at handling vast amounts of data, enabling swift and accurate insights. Finally, with sophisticated technology and real-time processing, financial transactions demand instant and continuous fraud detection. Employing real-time fraud detection mechanisms allows every transaction to be screened promptly, distinguishing between fraudulent and non-fraudulent activities. By utilising automated fraud detection, organisations can enhance their ability to swiftly identify and prevent fraudulent transactions, safeguarding their financial integrity and protecting their customers from potential harm.

1.2 Fraudulent Transaction

A fraudulent transaction is one where unauthorised access occurs, resulting in the withdrawal of funds from an individual's account or unauthorised use of payment information. The transaction may lead to the victim experiencing financial losses, losing personal asset, or compromising their personal information (Shinde et al., 2021).

Levi (2009) define fraud as “the obtaining of goods and/or money by deception”. Ramamoorti (2007) define fraud more elaborately as “a human endeavour, involving deception, purposeful intent, intensity of desire, risk of apprehension, violation of trust, rationalisation, etc.” However, the meaning and definition of fraud may change, depending upon the specific statute in which the word appears (Podgor, 1999).

Ahmed et al. (2016) have conducted a study where authors consider fraudulent transaction detection as anomaly detection problem where anomaly detection problem is defined as : “An anomaly is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism”.

In fraudulent transactions, scammers typically exploit the trust of victims or identify loopholes in the financial system to take advantage of vulnerabilities, engaging in unlawful activities. Some examples of fraudulent transactions include:

1. Credit card fraud: In this case, fraudsters use the victim's credit card information, including the credit card number, expiry date, and CVV number, to make purchases or withdraw money.
2. Identity theft: Fraudulent activities involve stealing personal identification numbers such as social security numbers or gaining access to login credentials for online banking, allowing perpetrators to impersonate individuals for financial gain.
3. Online Scams: Deceptive activities are conducted by fraudsters to obtain sensitive information or manipulate individuals into transferring money. For example, they may gain control of someone’s social media account and ask people directly connected to the individual to send money or provide personal sensitive information.
4. Check Fraud: Altering the check amount or unauthorised withdrawal of money from a victim's account.
5. Phishing: Collecting personal sensitive information by sending internet links or posing as a trustworthy entity in electronic communication.

1.3 Artificial Intelligence, Machine Learning and Deep Learning

The terms "artificial intelligence," "machine learning," and "deep learning" are extensively mentioned in this paper, with deep learning techniques being utilised in the project. To gain a comprehensive understanding of deep learning, it is essential to grasp the concepts of artificial intelligence and machine learning and comprehend the relationships and distinctions between these three terms, which will be briefly discussed here.

Artificial intelligence: Artificial intelligence (AI) is an emerging field that implements automated processes that detect and deter financial fraud. In AI, the objective is to create agents, computer programs, or hardware systems that can perform tasks related to these abilities. AI researchers focus on developing models or algorithms to simulate human-like cognitive abilities to carry out intelligent tasks. This involves designing systems that can reason, plan, solve problems, understand language, and learn from data, aiming to replicate and harness human-like cognitive abilities (Brey & Søraker, 2009). In addition, AI encompasses the intelligence displayed by machines, enabling them to sense their environment and make informed decisions to optimise the likelihood of achieving specific objectives (Ongsulee, 2017). It identifies emerging trends, makes predictions, and offers valuable insights through data accumulation and learning, resulting in improved workflow efficiency, faster response times, cost reduction, and enhanced client experiences (Zohuri & Rahmani, 2023). Machine learning and deep learning are the two critical areas of artificial intelligence. The background and concepts of machine learning and deep learning are now briefly discussed.

Machine Learning: Machine learning (ML) is a subfield of AI that provides computers with the capacity to learn and make predictions or decisions without being explicitly programmed (Ongsulee, 2017). It learns by extracting patterns and establishing relationships from the data to build a generalised model. The ML model can predict or make decisions for new and unseen inputs. Machine learning arises naturally from the convergence of computer science and statistics. Computer science focuses on building machines to solve complex but manageable problems. In contrast, statistics is related to extracting meaningful insights from data, aiming for a certain degree of reliability. Machine learning aims to enable computers to learn to program themselves from past experiences and initial input rather than manually creating computer programs. In addition to concluding data, ML also deals with algorithms, computational architectures, storage, data capture, indexing, merging techniques, and retrieval to effectively and efficiently manage and process data (Mitchell, 2006). The general steps involved in machine learning are described in steps 1 to 5 and in the flowchart Figure 1-1 (Alhaidari et al., 2022).

Machine learning working process:

Step 1: Collect the dataset.

Step 2: Split the dataset into training and testing sets.

Step 3: Train the model using the training dataset.

Step 4: Use the testing dataset to evaluate the model.

Step 4.1: Calculate the performance.

Step 4.1: Is the performance satisfactory? (Yes/No)

If the answer is Yes,

then go to Step 5.

If the answer is No,

Change the ML parameters and hyperparameters.

Go back to Step 3.

Step 5: Stop and save the model.

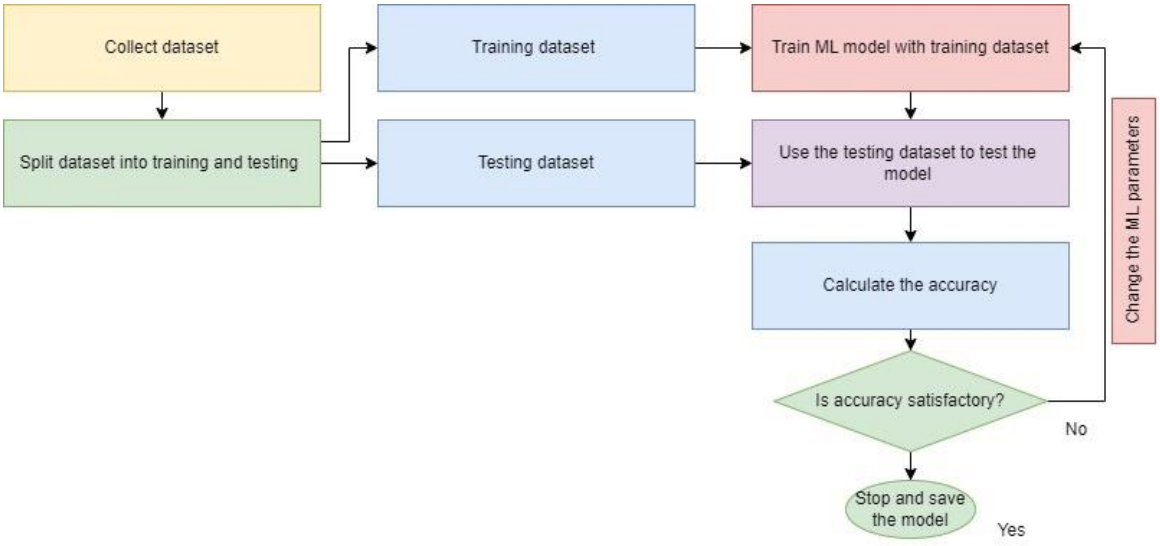


Figure 1-1 How Machine Learning Works.

Machine learning is broadly divided into three categories: supervised, unsupervised, and reinforcement.

Supervised learning involves training a model with labelled data, where the input is paired with respective output labels (Kotsiantis et al., 2006). The model is first trained using a training dataset, where it learns to identify insights or patterns from the data. It then generalises those patterns to make predictions on new, unseen data. Examples of supervised learning include linear regression, logistic regression, random forest, support

vector machines, K-nearest neighbour, neural networks, and Naïve Bayesian (NB) (Akinsola, 2017).

Unsupervised machine learning involves training a model without pre-defined labelled data. The model learns to find patterns, structures, and relationships within the data without guidance. Examples of unsupervised learning include clustering and anomaly detection. In anomaly detection, unusual patterns are detected in the data. K-mean clustering and principal component analysers are examples of unsupervised machine learning algorithms.

In reinforcement machine learning, an agent interacts with an environment through trial and error. The agent receives rewards for positive outcomes and is penalised for negative outcomes based on feedback from the environment (Barto & Dietterich, 2004). Examples of reinforcement learning include game-playing and robotics. In game playing, agents learn by exploring different moves, receiving rewards for winning and penalties for losing to improve their performance. In robotics, reinforcement learning can be applied to teach robots to perform tasks or navigate environments by adjusting their actions based on feedback.

The machine learning models that are commonly used are described below.

Logistic regression:

Logistic regression is the probability of an event occurring based on the given dataset called independence variables. The sigmoid or logistic function is used to model the relationship between the independent variable and the probability of the events. The sigmoid function generates the values between 0 and 1 for any real number (Awoyemi et al., 2017).

Random forest:

A random forest is the ensemble of a collection of decision trees to get a more accurate result. It works on different multiple decision tees on different subsets of the data and aggregates their output (Xuan et al., 2018). To reduce overfitting and generalisation of models, trees are selected randomly. Then the output of the trees is averaged for the regression problem or voted for a majority to solve the classification problem.

Support vector machine:

Support vector machine is a technique in which a classification problem is solved by a plane called a hyperplane between classes. Using the kernel function, SVM can solve the non-linearly separable data. It is used widely due to its strong generalisation capacity (Gyamfi & Abdulai, 2018).

K-nearest neighbour:

The k-nearest neighbour method is used in classification and regression problems. First, the labelled data point (training set) is segregated into K number of categories. When a new point of data is arrived, the new point of data is categorised based on the shortest distance from the previously defined categories in the classification problem. For the regression problem, the value of the new data point is predicted based on what is nearest to all the data points of the training set and by averaging the value, the final prediction of value is made (Tran & Dang, 2021).

Deep learning: Deep learning (DL) is a machine learning method that enables computers to gather knowledge from experience and realise the world using a series of interconnected layers of processing units (Bengio et al., 2017a). These layers extract and modify features in a non-linear processing unit. The output of one layer serves as input for the next layer, creating a cascading effect. Additionally, deep learning algorithms can acquire various levels of representations, each corresponding to different degrees of abstraction. These levels of abstraction form a hierarchical structure of concepts (Kim, 2016; Zhang et al., 2018). Deep learning is crucial in various fields of AI applications and services. It performs analytical tasks and physical tasks without human assistance. It is used in natural language processing, digital aid, computer vision, image recognition, anomaly detection and cutting-edge technologies such as driverless cars.

Deep learning uses various methods: Artificial Neural Networks, Deep Belief Networks, Convolutional Neural Networks, Autoencoders, and Recurrent Neural Networks (Raghavan & Gayar, 2019). These DL techniques are popularly used to detect and prevent fraudulent transactions. Each technique is described below in brief.

1. **Artificial Neural Networks (ANN):** Artificial Neural Networks are the fundamental building blocks of deep learning. The human brain's neural structure inspires them and consist of interconnected nodes organised into layers. Each node (also called a neuron) in a layer receives input, processes it using a non-linear activation function, and produces an output. ANNs are used for various tasks such as classification, regression, and pattern recognition. Neural networks learn through a process called "training." During training, a neural network updates its parameters (weights and biases) based on the input data and the corresponding target output. Guresen and Kayakutlu (2011), Shenvi et al. (2019), and Azhan and Meraj (2020) used artificial neural networks to detect electronic transaction fraud.

2. **Deep Belief Networks (DBN):** Deep Belief Networks are generative models with multiple layers of hidden units. They are typically used for unsupervised learning tasks like feature learning and dimensionality reduction. DBNs use a greedy layer-wise pre-training approach followed by fine-tuning using backpropagation (Hinton, 2009). Raghavan and Gayar (2019) also use this technique for fraud transaction detection.

3. **Convolutional Neural Networks (CNN):** Convolutional Neural Network (CNN) is a specialised type of Artificial Neural Network with multiple layers of neurons. The layers comprise convolutional, pooling, and fully connected layers (Albawi et al., 2017). Convolutional Neural Networks are primarily used for computer vision tasks, such as image recognition and object detection. CNNs use convolutional layers with learnable filters (kernels) that slide over the input image to detect various patterns and features. They can learn hierarchical representations of visual features, making them highly effective for image-related tasks. Raghavan and Gayar (2019), Zhang and Huang (2020), and Naby et al. (2021) investigated the performance of CNN in detecting fraudulent transactions.

4. **Autoencoders:** An autoencoder neural network is an unsupervised learning technique that utilises backpropagation for training. During this process, the network sets the target values to match the input values, effectively attempting to reconstruct the input data from the encoded representation (Ng, 2011). Autoencoders are unsupervised neural networks used for dimensionality reduction and feature learning. They consist of an encoder that

compresses the input data into a latent representation and a decoder that reconstructs the input data from the latent representation. Autoencoders are used for data denoising, anomaly detection, and data compression. Fraud transactions were identified by Mubalaike and Adali (2018), using an autoencoder.

5. **Recurrent Neural Networks (RNN):** A recurrent neural network (RNN) is a type of neural network that incorporates feedback connections and recursive backpropagation. Its structures vary, encompassing fully connected, partially connected, and multi-layered feedback networks, all featuring separate input and output layers (Medsker & Jain, 2001). Recurrent Neural Networks are designed to handle sequential data, such as time series and natural language. RNNs have loops that allow information to persist across different time steps, making them suitable for tasks that require memory or context. However, traditional RNNs suffer from the vanishing gradient problem, which hinders long-range dependencies. This led to the development of more advanced RNN variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). Benchaji et al. (2021) experimented on credit card fraud detection using LSTM recurrent networks.

6. Long Short-Term Memory Networks (LSTM): Long Short-Term Memory Network is a special kind of Recurrent Neural Network (RNN) used to handle long-term dependency in sequential data. LSTM networks overcome the exploding and vanishing gradient problem by maintaining two separate states. One state is known as the long-term state, and the other state is known as the short-term state. LSTM networks have been used in different applications such as natural language modelling, speech recognition, and time series prediction and forecasting. The model is used to learn long-term sequence dependencies.

LSTM networks comprise memory cells whose function is controlled using three gates: input gate, forget gate and output gate.

1. **Memory Cell:** The memory cell acts as a storage unit of LSTM. It manages the cell state by storing and updating its state.

2. **Gates:** LSTM consists of three control gates to manage the flow of information: forget gate, input gate, and output gate (Pulver & Lyu, 2017).

a) **Forget gate**: The forget gate determines which part of the cell state to remove. It gets the current input and previous hidden state as input passes these values to a sigmoid activation function and generates output for the forgotten factor for each element of the cell state.

b) **Input gate:** The input gate controls which part of input can be added to the cell state. The current input and the previous hidden state pass through the input gate. Two sub-gets process these values: i) the sigmoid function (input gate) and ii) the tanh function (input node), which produce a value between 0 and 1 for each element of the input.

c) **Output Gate:** This gate produces the output of the cell at a given time step. First, it takes the current input and the previous hidden state and produces a value between 0 and 1 for each input element, then, the values are multiplied by the cell state to generate the output.

3. Hidden State: This state serves as a short-term memory for the cell. It takes the current input, the previous hidden state and the output of the forget and input gates. This gate is updated at each step, passing a value to the next step.

During the forward pass, the LSTM networks take the input as a sequence and process the sequence step by step. The input is combined with the previous hidden state and controlled by gates to determine how the current cell state and hidden state should be updated.

LSTM parameters are updated in the training phase using backpropagation and gradient descent against the loss function. Optimiser is used to update and optimise the parameters. An LSTM (Long Short-Term Memory) model's training process involves feeding data sequences through the network in batches. During each iteration, the model makes predictions, compares them against the actual values using a loss function, and adjusts its internal parameters (weights and biases) through backpropagation and gradient descent to minimise the loss. This process is repeated over multiple epochs, gradually improving the model's prediction ability. Validation on a separate dataset ensures the model generalises well, and once satisfactory performance is achieved, the model is tested on unseen data to assess its real-world effectiveness.

Key Components in Deep Learning:

The key components in deep learning can be categorised into parameters and hyperparameters. These two terms are discussed below.

Parameters:

In deep learning, a parameter refers to a set of values that the model's networks learn from the training data. These values are associated with the network's connections (weights) and individual neurons (biases) (Dongare et al., 2012). Parameters are adjusted during the training process to minimise the loss function, allowing the network to make accurate predictions on new, unseen data. In CNN, the parameters include:

1. *Weights:* Weights are numerical values associated with the connections between neurons in the network. Each weight represents the strength and influence of the input from one neuron to another. These weights are adjusted during training to optimise the model's performance (Grossi & Buscema, 2007).

2. *Biases:* Biases are constant values associated with each neuron (except for the input layer) in the network (Buscema, 1998). They are added to a neuron's weighted sum of inputs before passing through the activation function. Biases provide flexibility to the model and allow it to learn non-zero intercepts.

Hyperparameters:

Hyperparameters are the configuration settings of a deep learning model that are set before training. They are not learned during training, unlike the model's parameters. Examples of hyperparameters include the learning rate, the number of hidden layers, the number of nodes in each layer, batch size, and activation functions. Choosing appropriate hyperparameters is crucial for the model's performance and generalisation ability.

The hyperparameters of CNN are discussed below.

1. *Nodes (Neurons)*: Nodes, also known as neurons, are the basic units in a neural network. They receive input data, apply an activation function to it, and produce an output propagated forward to the next layer in the network.

2. *Hidden Layers:* Hidden layers are between a neural network's input and output layers. They play a crucial role in learning complex patterns and features from the data.

The number of hidden layers and the number of nodes in each hidden layer are hyperparameters that need to be determined during the model design.

3. *Filters (Kernels):* Filters are small learnable matrices used in convolutional layers of CNNs. They slide over the input data (e.g., images) and perform convolution operations, detecting specific patterns and features. Multiple filters in a layer learn different features, which are combined to form higher-level representations.

4. *Activation function:* Filters are small learnable matrices used in convolutional layers of CNNs. They slide over the input data (e.g., images) and perform convolution operations, detecting specific patterns and features. Multiple filters in a layer learn different features, which are combined to form higher-level representations.

5. *Batch Size:* Batch size refers to the number of training examples used in a single forward and backward pass during the training process of a neural network. In deep learning, the training data is typically divided into mini-batches, and each mini-batch is processed through the network before updating the model's parameters (weights and biases) using backpropagation. The batch size is a hyperparameter that can impact the training process. A larger batch size may lead to faster training but requires more memory, while a smaller batch size may lead to slower training but can be more noise-resistant and generalise better.

6. *Learning Rate:* The learning rate is a hyperparameter that determines the step size at which the model's parameters (weights and biases) are updated during training. It controls the magnitude of parameter adjustments to minimise the loss function (measuring the model's performance). A high learning rate can cause the model to overshoot the optimal values, making converging difficult. On the other hand, a very low learning rate can result in slow convergence and possibly getting stuck in local minima. Selecting an appropriate learning rate is crucial for successful model training.

7. *Optimisation Function (Optimiser):* The optimisation function, also known as the optimiser, is a crucial component of the training process in deep learning. It determines how the model's parameters are updated based on the computed gradients during backpropagation. The optimiser aims to find the optimal set of parameters that

minimise the loss function. Some popular optimisation algorithms include Stochastic Gradient Descent (SGD), Adam (Adaptive Moment Estimation), and RMSprop (Root Mean Square Propagation).

Characteristics and Advantages of DL:

Deep learning methods can automatically learn complex patterns and features from raw data, eliminating the need for manual feature engineering (Najafabadi et al., 2015). They are highly scalable and can efficiently handle large amounts of data, making them suitable for big data problems. Deep learning techniques have achieved state-of-the-art performance in various domains, including image recognition, natural language processing, and speech recognition. With the advent of deep learning frameworks and advancements in hardware (e.g., GPUs and TPUs), training deep learning models has become more accessible and faster. Deep learning models can adapt and generalise well to new, unseen data, making them suitable for real-world applications.

The relation between AI, ML, and DL is shown in Figure 1-2 (Bengio et al., 2017b).

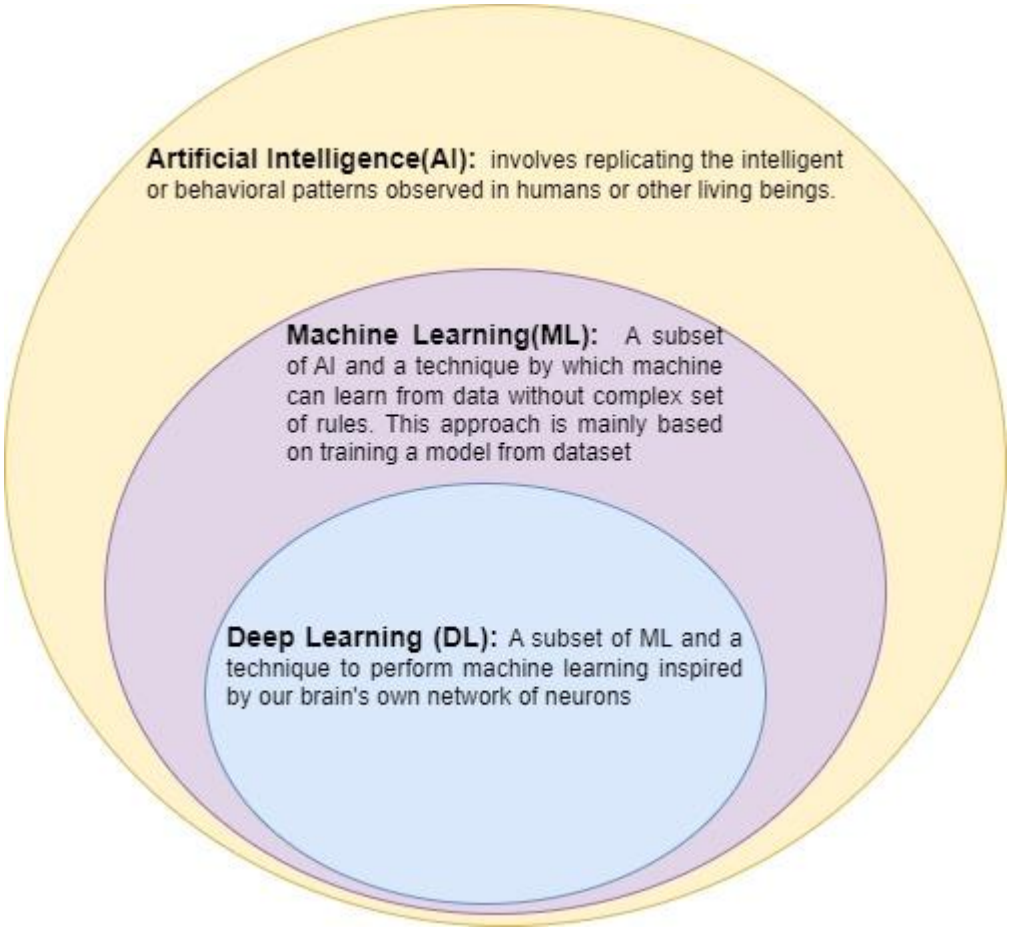


Figure 1-2 Relationship between Artificial Intelligence, Machine Learning, and Deep Learning.

1.4 Aims and Scope of the Research

This research aims at identifying and preventing fraudulent financial transactions using a modern machine learning technique, Deep Learning. Deep Learning (DL) is a machine learning (ML) technique that can learn patterns from the data, classify information and make predictions based on learning (Sharma et al., 2021). DL was chosen for its demonstrated superiority in various domains, its capacity to extract multiple levels of representation, incremental feature learning, and automatic feature engineering. DL's popularity is also increasing, driven by its remarkable accuracy when trained with vast amounts of data.

This project focuses on identifying the most popular DL methods for fraudulent transaction detection, the features used by those models for training and inference, and the hyperparameters tuned for boosting performance. An extensive literature review is conducted to find answers to these queries.

The selection of features plays an active role in the performance of DL algorithms. The features typically extracted by deep learning methods are not explainable. Many manual and automatic feature engineering methods extract features from input data, generating explainable features. Automatic feature engineering methods are less labour-intensive and less dependent on human expertise than manual feature engineering methods. To the best of our knowledge, the impact of using automatic feature engineering with deep learning for fraudulent transaction detection is yet to be studied thoroughly. The details of using automatic feature extraction to improve the performance of deep learning models are discussed in Chapter 4.

Various hyperparameters in deep learning, such as the number of layers, filters, filter/kernel size, pool size, learning rate, activation function, dropout rate, batch size, and optimiser selection in CNN, are crucial in achieving the best performance from the model. To the best of our knowledge, the impact of hyperparameter optimisation on the fraudulent transaction detection performance of DL models is yet to be analysed thoroughly. The details of using hyperparameter optimisation to improve the performance of DL models are discussed in Chapter 5.

1.4.1 Research Questions

In the context of the above discussion on DL models, features and hyperparameters, this project aims to answer the following research questions through a literature review:

RQ1. How is DL used to identify and prevent fraudulent transactions?

RQ2. What features of financial transactions are used to train the DL models?

RQ3. How many hidden layers and filters give optimal results for DL models used in fraudulent transaction detection?

RQ4. What is the best DL model for classifying fraudulent transactions?

1.4.2 Hypotheses

In the context of the above discussion on automatic feature engineering and hyperparameter optimisation, this project also aims to improve the efficacy of DL models by leveraging those techniques. Therefore, the following hypotheses have been constructed for this project, which are validated through experiments:

H1: Automatic feature engineering techniques significantly improve DL models' fraudulent transaction detection performance.

H2: Hyperparameter optimisation significantly enhances the efficacy of DL models in detecting fraudulent transactions.

1.4.3 Research Philosophy

The philosophical paradigm of the research project is positivism, which focuses on explaining existing knowledge of different DL models and their use in predicting anomalies in financial transactions. The project will contribute to understanding fraudulent transaction detection using DL models through scientific methods and observing the experimental results. The scientific experiment will be conducted as part of quantitative analysis to validate the hypotheses.

1.4.4 Objectives of the Research

The project aims to develop DL models to identify fraudulent transactions to reduce the losses incurred by businesses and financial institutions and increase customer confidence in online transactions. In particular, the objectives of the proposed research project are as follows:

1.4.4.1 Objective One

The first objective is to select and obtain transactional data for training and testing the DL models.

Sub-objectives:

1. To collect datasets from the open-source flats format and make them readily available for research.
2. To check the quality of data.

1.4.4.2 Objective Two

The second objective is to identify the typical features that discriminate between fraudulent and authentic financial transactions.

Sub-objectives:

1. To generate features from the dataset using suitable feature engineering techniques.
2. To evaluate the effect of feature engineering techniques on the DL models.

1.4.4.3 Objective Three

The third objective is to find the hyper-parameters that produce the best performance of the DL models.

Sub-objectives:

1. To optimise DL model performance, fine-tune hyper-parameters such as the number of layers, filters, filter/kernel size, pool size, learning rate, activation function, dropout rate, and optimiser selection.

1.4.4.4 Objective Four

The fourth objective is to train DL models, augmented with feature engineering and hyperparameter optimisation, to optimally learn the patterns in fraudulent transactions.

Sub-objectives:

1. To train the optimised DL models with features extracted from the collected datasets.
2. To detect fraudulent transactions using trained DL models.

1.4.4.5 Objective Five

The fifth aim is to compare the performance of the DL models with state-of-the-art models found in the literature to demonstrate the proposed DL models' efficacy.

Sub-objectives:

1. To construct the evaluation criteria for measuring the performance of models.
2. To present the results for a comparative study.

1.5 Structure of the Thesis

This thesis consists of six chapters.

Chapter 1 presents the introduction of the project, covering the background, aims and scope, research questions, hypotheses, philosophy and objectives of the research project.

Chapter 2 provides a systematic literature review of fraud detection methods using DL models intending to answer the research questions.

Chapter 3 outlines the research methodology and experiment design, which covers the data collection, data pre-processing, data pre-analysis, data splitting, model training and optimisation, model testing and evaluation, and provides a theoretical discussion of the models used for fraudulent transaction detection in this project.

Chapter 4 illustrates the use of a feature engineering technique with a deep learning model and analyses the impact of that feature engineering technique on the performance of the deep learning model in detecting fraudulent transactions.

Chapter 5 demonstrates the use of hyper-parameter tuning for improving the performance of a deep learning model and analyses the improvement in fraudulent transaction detection.

Chapter 6 summarises the project findings, states the limitations of the study and offers future research directions for the study.

1.6 Conclusion

This chapter has presented the introduction of the project, with a discussion of the background, aims and scope, research questions, hypotheses, philosophy and objectives.

The following chapter presents a systematic literature review to contextualise the proposed study and to answer the research questions RQ1-RQ4 formulated in the current chapter.

Chapter 2. Literature Review

2.1 Introduction

A systematic literature review was carried out to contextualise the proposed study and answer the research questions formulated in Chapter 1. The literature review collects, reviews, and synthesises existing studies on fraudulent transaction detection methods using deep learning (DL) models. Although some literature reviews have been conducted on fraudulent transaction detection using DL methods, systematic literature reviews on fraudulent transaction detection using deep learning are sparse. This study attempts to provide a clear view of fraud detection mechanisms using deep learning for researchers and practitioners. Another goal of this chapter is to identify the current studies' limitations and provide a pathway for future research related to fraudulent transaction detection. The answers to the four research questions are derived from data synthesis. From the answers, readers will understand the current DL models used for fraudulent transaction detection and the future research prospects in this area. The research questions formulated to guide the literature review are restated as follows:

RQ1. How is DL used to identify and prevent fraudulent transactions?
RQ2. What features of financial transactions are used to train the DL models?
RQ3. How many hidden layers and filters give optimal results for DL models used in fraudulent transaction detection?
RQ4. What is the best DL model for classifying fraudulent transactions?

The following section describes the systematic literature review method followed in this chapter. Section 2.3 presents the statistics from data extraction and synthesis. Section 2.4 presents the answers to the research questions derived from data synthesis. Section 2.5 provides concluding remarks.

2.2 Related Works

The survey by Hilal et al. (2022) presents a thorough review of the most popular and effective anomaly detection techniques applied to detect financial fraud, with a focus on highlighting recent advancements in the areas of semi-supervised and unsupervised learning. In the literature review, the authors discussed fraud as an anomalies problem

and defined anomalies, types of anomalies, challenges in anomaly detection, and a review in the perspective of datasets, such as Data Labels and output are explored. The authors also discuss how to measure performance in anomaly detection problems. The discussion covers different types of financial fraud, including credit card fraud, insurance fraud, money laundering, and other types of fraud. Various anomaly detection methods are explored, encompassing Supervised, Semi-supervised, and Unsupervised Methods. The literature review extensively discusses the pros and cons of fraud detection methods. However, the detailed selection of literature is not elaborately discussed.

In the review by Lim et al. (2021), the drawbacks of rule-based techniques for fraud detection were identified. The authors discussed data mining and machine learning algorithms; however, they did not provide a deep learning model for fraud detection.

Boutaher et al. (2020) illustrated the essential elements of identifying fraud, the prevailing fraud detection systems, challenges associated with banking-related frauds, and the current machine learning-based solutions in place. However, this review used a very small number of literature sources.

Pang et al. (2021) have conducted a literature review that identified the profile and challenges of anomaly detection. They provided a framework for categorisation and formulation of anomaly detection, reviewing a number of journal and conference papers on machine learning, deep learning, and data mining. The review also discusses future research opportunities to address the changes and, lastly, provides information on publicly accessible datasets.

West and Bhattacharya (2016) have conducted a comprehensive review of financial fraud detection, which includes computational intelligence techniques, finding research gap analysis, key issues, and challenges regarding financial fraud detection.

2.3 Review Method

A systematic literature review is an evidence-based approach to empirically carry out the research within the same context to answer predefined research questions and select, study, analyse, critique, and identify the research gaps to answer research questions (Kitchenham, 2007). Several articles and papers were selected from the ACM digital library, Scopus, IEEE Explore, and Web of Science databases to conduct the literature review. These databases are used popularly in engineering and artificial intelligence,

providing papers relevant to those fields. The systematic literature review in this chapter is conducted following the recommendations provided by Kitchenham (2007), as illustrated in Figure 2-1.

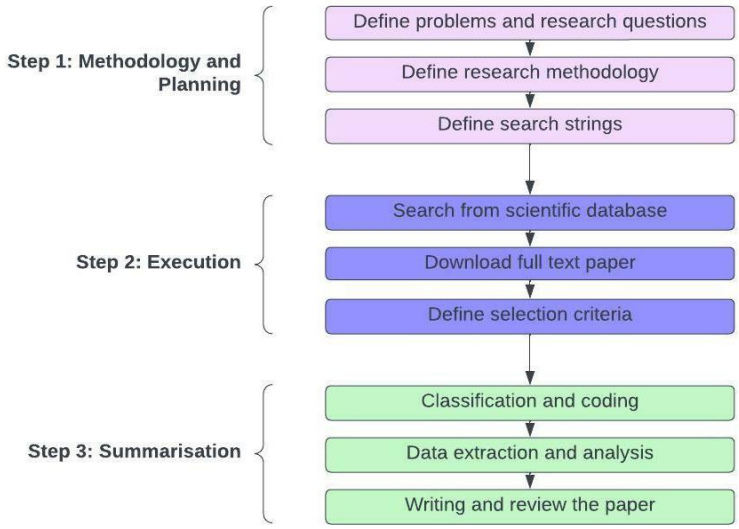


Figure 2-1 Methodology Used in the Systematic Literature Review.

2.3.1 Inclusion /Exclusion Criteria

We considered the title and abstract of each article according to the inclusion and exclusion criteria (see Table 2-1). Then, as a second filter, the full articles from the first filter were carefully read.

Table 2-1 Inclusion and Exclusion Criteria

No.	**Inclusion/ Exclusion Criteria**
IC1	Publication within the last 10 years (from March 2012 to March 2022)
IC2	Scope of the study: Engineering, Computer Science, Artificial Intelligence
IC3	Primary Studies (articles or journal or conference paper)
IC4	The paper that has discussed fraud detection techniques using DL
IC5	The paper contains information relevant to one or more research questions.
No.	**Exclusion Criteria**
EC1	Publications that have no full text have been excluded.
EC2	Duplicate papers have not been selected.
EC3	Publications that do not meet the inclusion criteria have been excluded.
EC4	The publications written in other than the English language have been excluded.

2.3.2 Study Selection

The study papers were selected through the following procedures:

1. Identification: Search strings were constructed using the keywords (see Table 2.2) targeting the title and abstract of the databases ACM, Scopus IEEE Explore, and Web of Science. The search strings were listed in Table 2.3, which were applied to the respective databases to select papers published from March 2012 to March -2022. Unfortunately, the search generated many irrelevant outcomes. To exclude irrelevant papers, the next steps were followed.
2. Filter: All papers from stage 1 were filtered using the exclusion and inclusion criteria. If any difficulties or doubts arose in choosing the papers, they were passed on to the next stage.
3. Eligibility: At this stage, the papers were finalised after text scanning, which was relevant to the research questions and satisfied the inclusion and exclusion criteria.
4. Data Extraction: Data from the selected papers was extracted to find the answers to RQ1-RQ4.

Table 2-2 Keywords

No	Description
1	deep learning
2	fraud detection
3	fraudulent

Table 2-3 Search String

Database	Search String
ACM DL	Title:((deep learning OR machine learning)) AND Title:((”fraud”) or (”fraud detection”) or (”Fraudulent”)) AND Abstract:((”deep learning” OR ”machine learning”)) AND Abstract:((”fraud”) or (”fraud detection”) or (”Fraudulent”)) ”filter”: Publication Date: (03/01/2012 TO 03/31/2022), ACM Content: DL
IEEE Xplore	(”All Metadata”:” deep learning” or ”machine learning”) AND (”All Metadata”:” Fraud Detection”)
Scopus	TITLE-ABS-KEY ((”deep learning” OR ”machine learning”) AND (”fraud detection” OR ”fraudulent”)) AND PUBYEAR ¿ 2011 AND PUBYEAR ¡ 2023 AND (LIMIT-TO (PUBSTAGE, ”final”)) AND (LIMIT-TO (DOCTYPE, ”cp”) OR LIMIT-TO (DOCTYPE, ”ar”) OR LIMIT-TO (DOCTYPE, ”cr”)) AND (LIMIT-TO (SUBJAREA, ”COMP”) OR LIMIT-TO (SUBJAREA, ”ENGI”)) AND (LIMIT-TO (LANGUAGE, ”English”)) AND (LIMIT-TO (OA, ”all”))
WOS	deep learning OR machine learning(Title) and fraud detection or ”Fraudulent” (Title) and deep learning OR machine learning(Abstract) and fraud detection or ”Fraudulent” (Abstract) and 2012-03-31 to 2022-03-31 (IndexDate)

2.4 Data Extraction and Analysis

The data was extracted from the selected papers and then analysed and classified according to the relevancy of addressing the questions to find out the answers to the research questions. Finally, a data extraction table (see Table 2-4) has been formulated to analyse the selected papers comprehensively.

Table 2-4 Data Extraction Table

No	Extracted Data	Description	Type
1	Study identification	Uniquely identifying the study	general
2	Reference for bibliography	Authors, title, publication year, and publication source	general
3	Study type	Journal article, Conference paper	general
4	The theories applied	How DL used to identify fraud detection	RQ1
5	Techniques for feature engineering	Description of features identification of financial transactions	RQ2
6	Parameter setup of models	What parameters will provide the optimal result	RQ3
7	Performance comparison	Description of DL models which performed better	RQ4
8	Result analysis	Description	general

2.4.1 Synthesis

In the initial selection, the titles and abstracts of some papers were not related to the context of the research questions but matched with the search string. Thus papers were selected considering the nature of the information concerning the research area and the ability to answer the research questions. A pie chart illustrating the number of articles retrieved using the search string is provided in Figure 2-2.

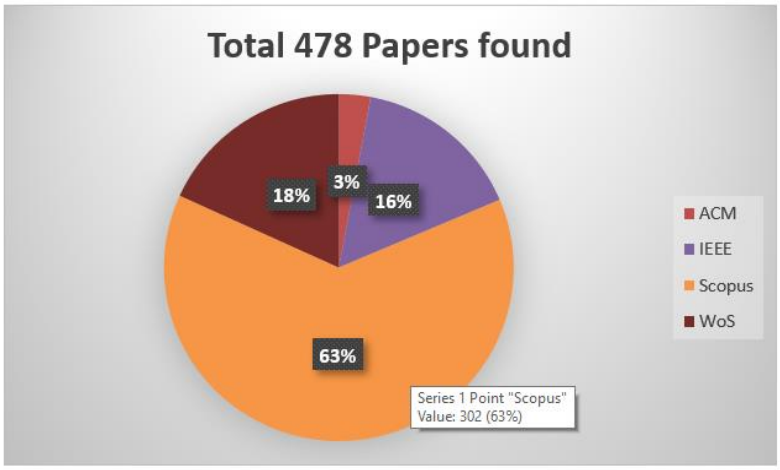


Figure 2-2 Papers Found using Search Engine.

Table 2-5 shows the number of papers found in different databases by matching the search strings. The number of initially selected papers found in the databases is presented in the second column. The third column shows the number of articles after manually scanning the title and abstract. The number of duplicate papers is shown in the fourth column, and the initial selection is shown in the fifth column after removing the duplicate papers. Lastly, the deep learning and full-text papers were selected for final inclusion. The selection process of the papers from the initial step to the final step is shown in Figure 2-3. In the primary selection from the four databases above, 481 papers were found by applying inclusion and exclusion criteria. After matching the title and abstract in full-text articles, 253 papers were selected. After removing the 32 duplicate papers, the initial selection stood at 221. Then, from these, 221 papers had the full text. Finally, 28 papers that dealt with deep learning models have been selected for the SLR.

Table 2-5 Number of Papers Selected using Selection Criteria

Source	Papers Found	Select Based on Title and Abstract	Duplicate	Initial Selection	Full Text with DL
ACM	14	12	0	12	0
IEEE	75	64	0	64	12
Scopus	302	111	16	95	9
WOS	87	66	16	50	7
Total	478	253	32	221	28

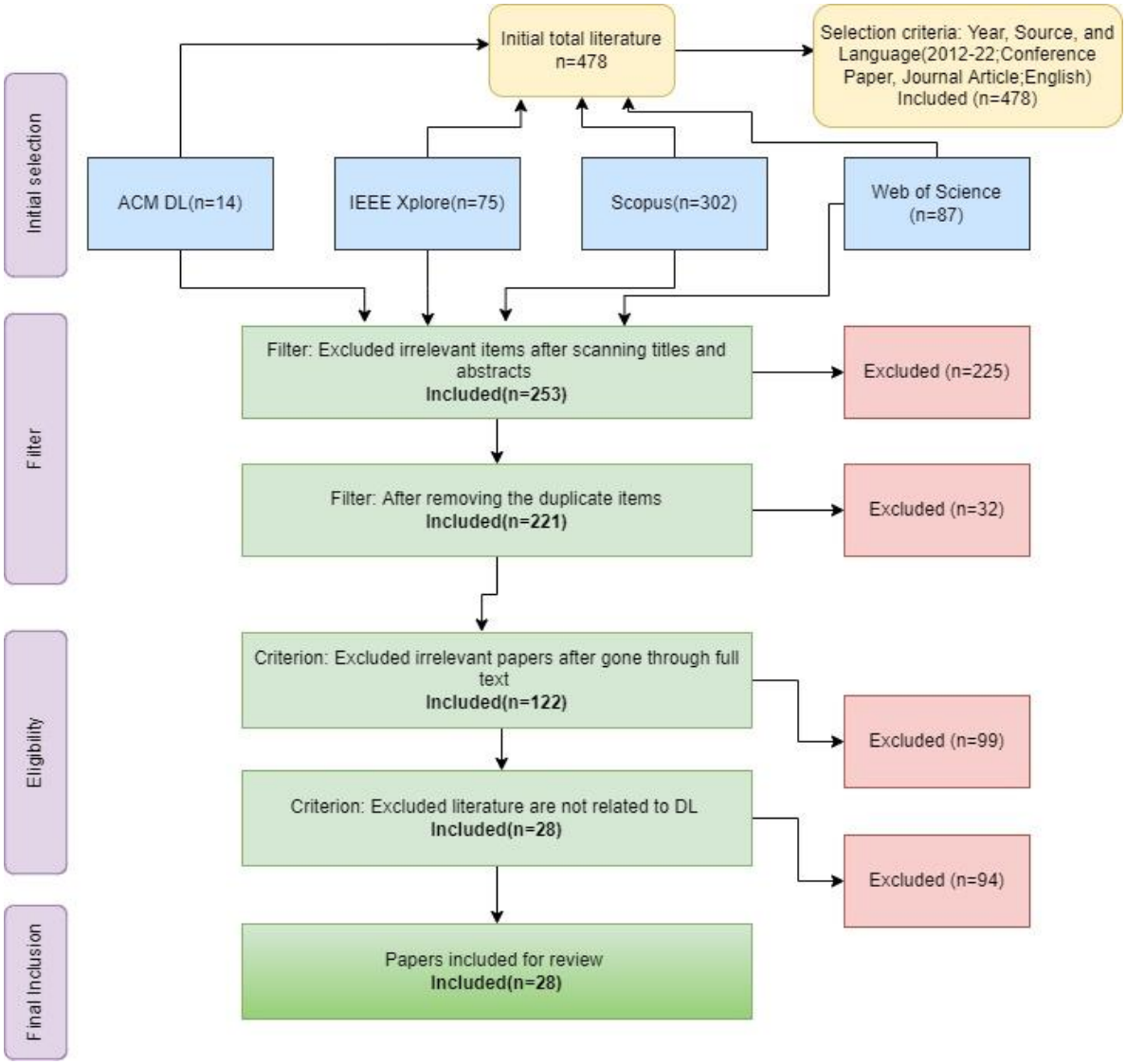


Figure 2-3 Flowchart for Narrowing Down Search Results.

Year-wise distribution of papers is shown in Figure 2-4. No paper was published

between 2012 and 2017. Among the selected papers, eight papers were published in each of the years 2020 and 2021. Only four of the selected papers were published in 2022, as papers were selected from only the 1st quarter of that year. On the other hand, only four selected papers were published in 2018 and 2019.

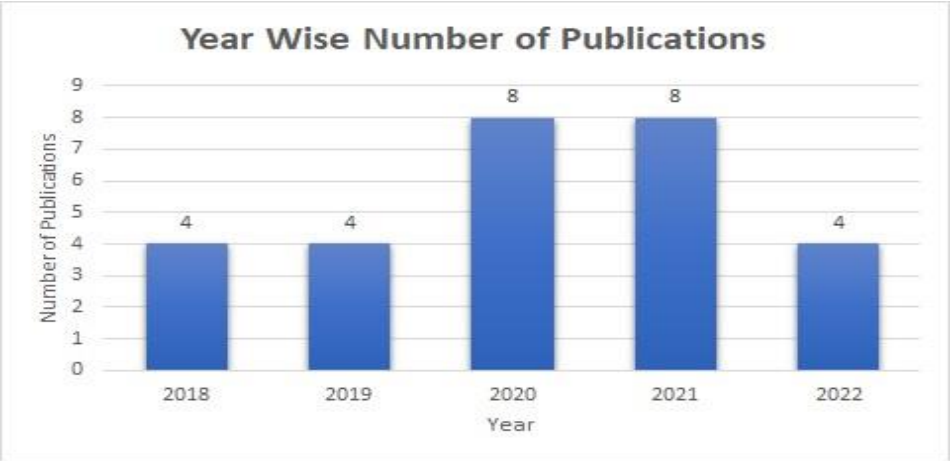


Figure 2-4 Year Wise Number of Publications.

2.5 Research Questions Results

Of the selected 28 papers, 23 are relevant to RQ1, seven are relevant to both RQ2 and RQ3, and 18 are relevant to RQ4, shown in Table **2-6**.

Table **2-6** also shows the frequency of papers found vs. the research questions. It is found that RQ1 is the most frequently investigated research question, which accounts for 82.14% of the total frequency, and RQ4 is the second most frequently investigated research question, which accounts for 60.71% of the total frequency, while RQ2 and RQ3 account for 25% of the total frequency.

Table 2-6 Data Extraction Table

RQ	Paper Reference	Frequency	Percentage
RQ1	(Alarfaj et al., 2022; Alharbi et al., 2022) (Benchaji et al., 2021) (Carrasco & Sicilia-Urbán, 2020) (Dang et al., 2021) (Ikeda et al., 2021) (Kewei et al., 2021) (Li et al., 2020) (Liu et al., 2020) (Liu et al., 2021) (Mubalaike & Adali, 2018; Naby et al., 2021) (Najadat et al., 2020) (Pillai et al., 2018) (Pratuzaite & Maknickiene, 2020) (Pumsirirat & Yan, 2018) (Sanober et al., 2021) (Shenvi et al., 2019) (Yinze, 2022) (Zhang & Huang, 2020) (Youssef et al., 2020) (Wang et al., 2021) (Smiles & Sasi Kumar, 2019)	23	82.14%
RQ2	(Alarfaj et al., 2022) (Babu & Pratap, 2020) (Carrasco & Sicilia-Urbán, 2020) (Pumsirirat & Yan, 2018) (Ikeda et al., 2021) (Kewei et al., 2021) (Liu et al., 2020)	7	25%
RQ3	(Alarfaj et al., 2022) (Babu & Pratap, 2020) (Benchaji et al., 2021) (Carrasco & Sicilia-Urbán, 2020) (Naby et al., 2021) (Kewei et al., 2021) (Zhang & Huang, 2020)	7	25%
RQ4	(Alarfaj et al., 2022; Alharbi et al., 2022) (Carrasco & Sicilia-Urbán, 2020; Ikeda et al., 2021) (Kewei et al., 2021) (Liu et al., 2020) (Liu et al., 2021) (Mubalaike & Adali, 2018) (Najadat et al., 2020) (Pratuzaite & Maknickiene, 2020) (Raghavan & Gayar, 2019) (Sanober et al., 2021) (Zhang & Huang, 2020) (Smiles & Sasi Kumar, 2019) (Ullah et al., 2022) (Wang et al., 2018) (Khazane et al., 2019)	17	60.71%

2.5.1 Finding the Answers to the Research Questions

2.5.1.1 RQ1: How *is DL used to identify and prevent fraudulent transactions?*

This subsection analyses papers linked to RQ1, i.e., the DL models used to detect fraudulent transactions. A total of 23 papers are found related to RQ1. Firstly, Alarfaj et al. (2022) investigated a fraud detection model for the European card benchmark dataset using three Convolutional Neural Networks (CNN) architectures. They pointed out that a CNN-based model with 20 layers was the best-performing method, with an accuracy of 99.72%. Alharbi et al. (2022) used the text2IMG conversion and inverse frequency methods to address the class imbalance problem in another study. They provide a new dimension to credit card fraud detection using computer vision techniques. They offer a future direction for converting other types of textual data into images, allowing for data classification with unique patterns. This study also proposed a new technique in this domain.

Benchaji et al. (2021) used the LSTM recurrent networks to incorporate the transaction sequences and applied the selective attention mechanism to improve performances. They also used the Uniform Manifold Approximation and Projection (UMAP) techniques to select the best features for improving prediction accuracy.

Carrasco and Sicilia-Urbán (2020) experimented with several deep networks to assess their capacity to reduce false alarms. The best setting was able to reduce false alarms by 35.16% with an ability to detect 91.79% of the fraud events. In the study by Dang et al. (2021), KNN (K Nearest neighbours), LR (Logistics Regression), DT (Decision Tree), RF (Random Forest), Deep Neural Networks, Adaptive Boosting (AdaBoost), and eXtream Gradient Boosting (XGBoost) were used to detect credit card fraud. In addition, they used two resampling techniques, SMOTE and ADASYN, in the data pre-processing stage before the training/test split to mitigate the data imbalance problem, resulting in above 99% accuracy for all ML algorithms.

Ikeda et al. (2021) used feature engineering with SVM, Isolation Forest (IF), and Autoencoder for fraud detection. They used the dataset from a European private bank for the research. They found that the performance of the Autoencoder model with selected features performed better than that built with raw data only.

The study by Kewei et al. (2021) used feature compression, mixed precision, and ensemble loss techniques to enhance the performance of their proposed deep learning-based models to detect fraud in financial transactions. Their model was trained on the dataset from Vesta Corporation, and this model was found to outperform traditional methods such as Bayes and SVM. Memory compression and hybrid precision reduce the model size by 15%, allowing faster model training. The proposed model digests large amounts of transaction data and makes better predictions in real-time. However, it treated data as an isolated entity and did not consider the sequence of transactions. A study performed by Li et al. (2020) used Full Centre Loss (FCL), which is meant to supervise deep representation learning. The FCL technique was compared with other state-of-the-art loss functions, and FCL was found to outperform others in terms of credit card fraud detection. The encoders were used in the study to extract the selected features from the datasets to reduce space, time, volume, and computational complexity. The DL model was used to represent the features in behavioural and visual word form. To enhance classification performance, they made a fusion with different features. Their approach used feature fusion using the weighted correlation method. Finally, sparse reconstruction errors were used to identify fraud in financial transactions. The proposed algorithm used important data characteristics to train the model effectively, producing a higher detection rate and reducing computational complexity. A combined structure of 1D-Conv and the

residual neural network (Res-net), named CCFD-net, was proposed by Liu et al. (2021). The datasets of e-commerce transactions of Vesta corporation were used in this study, found at https://github.com/zhenguonie/2021 CCFD Net. The k-fold cross-method was used to assess the different models' performance. The proposed CCFD-Net exhibited higher AUROC and F1 scores than traditional machine learning models.

Pumsirirat and Yan (2018) and Mubalaike and Adali (2018) used Auto-Encoder (AE) and Restricted Boltzmann Machine (RBM) that focused on fraud cases that cannot be detected based on the previous transaction history (supervised learning) and found that AE and RBM have produced a high AUC score and accuracy for bigger datasets.

Naby et al. (2021) used the OSCNN (Over Sampling with Convolution Neural Network) on the fraud dataset from the Kaggle.com website for fraudulent transaction detection. The experimental results suggested that OSCNN surpassed MLP and MLP + SMOTE by a large margin. Najadat et al. (2020) proposed methods, named BiLSTM-MaxPooling-BiGRU-MaxPooling based on Bidirectional Long Short-term memory (BiLSTM) and Bidirectional Gated Recurrent Unit (BiGRU) to detect credit card fraud. The proposed method was compared against the Naïve Bayes, Voting, Random Forest, Logistic Regression (LR), Decision Tree, and Ada Boosting, using the IEEE-CIS Fraud Detection dataset, and was found to outperform all traditional methods.

A study performed by Pillai et al. (2018) experimented with various parameters of the Multilayer Perceptron (MLP) using the ULB (Université Libre de Bruxelles) dataset. The study revealed that MLP performed better with the logistic activation function than the tanh function. The sensitivity is the lowest for the identity activation function as it does not transform the input value.

Generalised Method of Moments, K-nearest Neighbours, Naïve Bayes Classification, and Deep Learning methods have been used in the study by Pratuzaite and Maknickiene (2020), which showed that Deep Learning using neural networks achieved the highest recognition accuracy, which was 99.9%.

Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), Logistic Regression (LR), Support Vector Machines (SVM), K-nearest Neighbours (KNN), and Neural Networks were used in the study by Sanober et al. (2021), and they listed the performance of each model in terms of accuracy, specificity and F1 score, where RF performed better

than other models. A study conducted by Shenvi et al. (2019) showed that the performance of the Neural Networks model is better than that of the RF and SVM models. However, training a Neural Networks model is more complicated than training an SVM model. Both oversampling and undersampling were used to balance the dataset's skewness, where the undersampling method performed better than the oversampling method. However, the authors suggested that the oversampling method is ideal for training a model since no data is deleted from the sample.

Yinze (2022) reviewed some existing deep learning fraud detection techniques, including Neural Networks, CNN, and Autoencoders. This work also identified some challenges of fraud detection, which are the use of diverse metrics, dissimilar data sets, and the survivor bias problem. Zhang and Huang (2020) investigated the performance of CNN and Autoencoder deep learning techniques in detecting fraudulent transactions. Their experiment used raw and Libre de Bruxelles (ULB) transaction datasets. The study found that undersampling with CNN demonstrated better performance than Autoencoder. The CNN deep learning method could detect fraud in large, complex, raw-input datasets. On the other hand, the Autoencoder performed relatively better with a large number of parameters. Random undersampling performed better than SMOTE on both models, as the first method dealt with real datasets, whereas SMOTE generates synthetic virtual data. However, the overfitting problem occurs with random undersampling methods.

Youssef et al. (2020) built a novel deep learning-based model that used a Continuous/Discrete Rule extractor via a Decision Tree algorithm to extract rules for detecting fraudulent transactions. Their experiment used feature aggregation and re-sampling to highly imbalanced datasets.

Wang et al. (2021) proposed two improvements to Graph Neural Networks (GNNs), inspired by their prior success in detecting fraudulent activities. The first improvement was a graph transformation method to embed the structural information, and the second improvement was a graph pre-training strategy to utilise more unlabelled data. Their experiment on large-scale industrial datasets proved the efficacy of their proposed improvements. Smiles and Sasi Kumar (2019) investigated the performance of their proposed model, SMOTE regularised Deep Autoencoders (SRD Autoencoders), and compared it against those of Ensemble Classifiers, Bagging, and Boosting techniques for

detecting fraud. The experimental results suggested that the proposed model outperformed the other models.

2.5.1.2 RQ2: What features of financial transactions are used to train the DL models?

This section presents an analysis of the selected papers relevant to RQ2, i.e., the features of financial transactions to be used to train deep learning models.

Alarfaj et al. (2022), Babu and Pratap (2020), and Carrasco and Sicilia-Urbán (2020) used Convolutional Neural Networks (CNN) techniques for the detection of fraud using the dataset of European cardholders. The features were extracted by the convolutional layers of CNN in this study. Hence manual feature engineering was not needed to extract the features from the dataset. In the research papers by Pumsirirat and Yan (2018), Carrasco and Sicilia-Urbán (2020), Ikeda et al. (2021), and Liu et al. (2020), an Autoencoder model was used to extract features. Autoencoders first encode input data and reconstruct it from the encoding. Through this process, autoencoders learn the useful features of the input data and are able to produce a compressed representation of the features. Ikeda et al. (2021) found that the performance of the autoencoder model with selected features performs better than the ones that are built with raw data only.

A study by Kewei et al. (2021) used feature compression techniques using the dataset from Vesta Corporation. They found that memory compression and hybrid precision reduce their model size by 15%, making the model's training faster. As a result, their proposed model can digest a considerable amount of transaction data and make better predictions in real-time.

2.5.1.3 RQ3: How many hidden layers and filters give optimal results for DL models used in fraudulent transaction detection?

The section provides a summary of the analysis pertaining to RQ3, which is about setting up the optimal parameter for the DL models. Seven of the selected papers are relevant to RQ3.

Alarfaj et al. (2022) found that the CNN model with 20 layers performed best. Babu and Pratap (2020) dropped 50% of the neurons from the second layer and 20% of the neurons from the first layers of the CNN model. They found that dropping neurons from max pooling layers improved the accuracy of the model’s performance from 95.93%

to 99.62%. In the study by Benchaji et al. (2021), a six-layer LSTM model was used for fraud detection. In the attention layer of the model, the global dependencies were identified from the transaction sequence, which focused on the most relevant data items for detecting fraudulent transactions. As part of their study, Carrasco and Sicilia-Urbán (2020) varied the number of epochs, batch size, learning rate and the depth of hidden layers. They noted that the optimal configuration of the MLP2OH128H918 captured 91.79% of fraudulent cases (8.21% misclassification rate) while achieving an alert reduction rate of 35.16% (threshold = 0.1).

The study by Naby et al. (2021) evaluated the performance of MLP with hyper-parameters: number of epochs = 20, batch size = 1000. They also evaluated the performance of the MLP model by varying the number of Epochs to 16, 30 and 50. It was observed that the accuracy did not change significantly with the changes in the number of epochs. However, accuracy increases when the data is oversampled with the SMOTE technique. In this same study, the authors evaluated the accuracy of the OSCNN model for fraud detection by varying the number of epochs. Unlike MLP, the accuracy of OSCNN improved with the increase in the number of epochs.

The grid search method is used in the study by Kewei et al. (2021) to optimise the hyper-parameters using Nadam Optimiser. The optimal values were: learning rate = .0001, number of epochs = 7, and batch size = 2048. The study by Zhang and Huang (2020) used 50 epochs and an Adam optimiser with a learning rate .0005.

2.5.1.4 RQ4: What is the best DL model for classifying fraudulent transactions?

The analysis of the papers relevant to RQ4 is illustrated as follows.

Alarfaj et al. (2022) used three DL architectures based on Deep Belief Networks, Convolution Neural Networks, and Deep Autoencoders, where it was observed that the Convolutional Neural Network (CNN) with 20 layers is the top-performing model resulting in an accuracy of 99.72%.

Alharbi et al. (2022) used CNN with text2IMG conversion technique and inverse frequency method in class imbalance dataset for fraud detection. They found that the proposed CNN performed with an accuracy rate of 99.87%.

The performances of MLP, CNN, and DAE, fraud detection models, were evaluated by Carrasco and Sicilia-Urbán (2020). CNN was the best performer among the three DL models, with a recall rate of .78%.

Naby et al. (2021) have used OSCNN (Over Sampling with Convolution Neural Network) for fraud detection using the Kaggle dataset and found that CNN+SMOTE (OSCNN) performed better with an accuracy rate of 98.9% with comparison to CNN+KNN, CNN, and Deep neural networks.

Kewei et al. (2021) used mixed precision, feature compression, and ensemble loss to increase the performance of a deep learning-based model. They found that their proposed deep learning-based model outperformed Naïve Bayes and SVM with an AUC score of 0.910 and an accuracy rate of 0.958.

Liu et al. (2020) used encoders with deep networks to detect fraudulent transactions. They observed that their proposed model would effectively learn the important features from the data. As a result, the proposed model showed good performance in fraud detection, and the computation complexity was comparatively low. In the study of Liu et al. (2021), a hybrid architecture of 1D-Conv and the residual neural network (Res-net), named CCFD, performed better than most of the Classic Machine learning models.

A comparative study by Mubalaike and Adali (2018) observed that Stacked Auto-Encoders (SAE) is the best performer, followed by Random Forest and KNN.

The study performed by Najadat et al. (2020) proposed BiLSTM-MaxPooling, and BiGRU-MaxPooling, which performed better than Naïve Bayes, Voting, Random Forest, Logistic Regression (LR), Decision Tree, Ada Boosting, with the AUC 91.37%.

The study by Pratuzaite and Maknickiene (2020) found that deep learning with neural networks achieved the highest recognition accuracy of 99.9% compared to the Generalised method of moments, K-nearest neighbours, and Naïve Bayes classifiers.

Raghavan and Gayar (2019) showed that an ensemble of Random Forest, SVM, CNN, and with a majority voting classifier, outperformed individual classifiers in respect of Area under the ROC Curve (AUC), Mathews Correlation Coefficient (MCC), and Cost of Failure.

Sanober et al. (2021) used Neural Networks, Support Vector Machine (SVM), Naive Bayes (NB), Logistic Regression (LR), Decision Tree (DT), K-nearest Neighbours (KNN), and Random Forest (RF) models. They found that RF performed best among all models based on accuracy, specificity, precision, and F1-score.

A study by Zhang and Huang (2020) used CNN and Autoencoder for fraud detection and found that CNN performed better than Autoencoder. Smiles and Sasi [27] have investigated the performance of fraud detection using Reduction algorithms such as PCA, tSNE, and UMAP, and Synthetic Minority Oversampling Technique, using the datasets from Kaggle.com, and their proposed model achieved better results than the GRU, LSTM, SVM, KNN, and ANN classifiers.

Ullah et al. (2022) used the Layer-wise Relevance Propagation (LRP) explainability technique in Deep Learning Neural Networks for credit card fraud detection. Their proposed model's performance was as follows: accuracy .9991, precision 0.6732, recall .9520, specificity .9992 and F1score .7866.

Wang et al. (2018) used DeepFD, a novel deep-structured learning model that simultaneously stores the non-linear graph structure and user behaviours. The result showed that their proposed model outperformed the state-of-the-art baselines.

A study by Khazane et al. (2019) used Embedding Graphs in financial fraud detection, and the model scored a prediction AUC of 0.96.

Summary of the literature review: In this literature review, we examine several research papers focusing on fraud detection using different approaches, including deep learning, neural network architectures, and machine learning methods. By analysing the pros and cons of some literature from reviewed papers (see Table **2-7**), the critical findings are listed in Table 2-8.

Table 2-7 Pros and Cons of the Reviewed Literature

Reference of Papers	**Pros**	**Cons**
(Alarfaj et al., 2022)	Provides a clear description of the dataset and uses multiple machine learning techniques, including ensemble learning and deep learning. Describes performance evaluation clearly.	Lacks information in data pre-processing. Lack of logical coherence and connectivity between sections and subsections, making it difficult to establish relationships between them.
(Benchaji et al., 2021)	Uses PCA, t-SNE, and UMAP for feature selection and dimension reduction. The LSTM model explores long-term dependencies in transaction sequences. The attention layer enables the model to focus on specific features.	Plots accuracy and recall curves for comparison. However, it lacks performance comparison on training or test datasets.
(Carrasco & Sicilia-Urbán, 2020)	Explores multiple neural network architectures, improving accuracy, reducing false positive rates, and minimising human labour costs.	Uses a dataset from a Spanish payment processing system but has not tested on other datasets, limiting its applicability to other fraud detection domains.
(Dang et al., 2021)	Utilises ML and Deep Reinforcement Learning (DRL) for CCF detection. Explores resampling approaches: SMOTE and ADASYN	Lacks addressing feature engineering and preprocessing techniques could enhance the efficacy of the CCF system.
(Kewei et al., 2021)	Uses feature engineering and memory compression in the preprocessing stage. Employs binary cross entropy and focal loss ensemble for deep learning model—Optimises hyperparameters with grid search.	This paper does not explain the data split for training and testing nor compare performance with state-of-the-art models.
(Li et al., 2020)	Utilises representational learning and a new loss function (full centre loss) for deep representational learning.	Discusses context and experimental results but lacks information on the practical implementation of the proposed project.

(Liu et al., 2020)	The algorithm uses four indicators (accuracy rate, recall rate, FM, and ROC chart) to measure experimental result quality. Demonstrates good predictive ability regardless of data distribution.	Lacks details on the algorithm used and does not employ hyperparameter optimisation techniques.
(Liu et al., 2021)	Deep learning models, like CCFD-Net, show stronger effectiveness and stability than traditional machine learning methods.	CCFD-Net requires substantial computational resources and longer training time due to its seventy-six-layer model.
(Mubalaike & Adali, 2018; Naby et al., 2021)	Utilises OSCNN with better accuracy than Multi-level Perceptron. Addresses data imbalance problem.	Lacks a validation process and does not provide metrics such as F1 score, Specificity, or AUC.
(Najadat et al., 2020)	Compares machine learning classifiers with the proposed model, achieving a better score.	Lacks a summary of key findings and directions for future research.
(Pratuzaite & Maknickiene, 2020)	Compares deep learning with a generalised method of moments, KNN, and Naïve Bayes classification	Does not compare model performance using training and validation datasets.
(Pumsirirat & Yan, 2018)	Addresses unsupervised learning techniques for fraud detection. Uses Autoencoder and Restricted Boltzmann Machine.	Does not compare performance with existing fraud detection algorithms.
(Shenvi et al., 2019)	Claims better accuracy using deep neural networks and sampling techniques.	Only provides an accuracy score, neglecting other measurement metrics. The model is not tuned for optimal results.
(Zhang & Huang, 2020)	Tests CNN and Auto-encoder models on different datasets. Samples data using undersampling and oversampling methods, showing better performance with undersampled datasets.	The performance of the model varies significantly between the two datasets, indicating a lack of generalisation.

(Youssef et al., 2020)	Uses a novel approach called Continuous/Discrete Rule Extractor via Decision Tree (CRED).	Increases accuracy and recall but does not improve precision. Does not provide analysis, discussion, or future implications of the project
(Wang et al., 2021)	Introduces a graph transformation method to convert multi-entity graphs into smaller, single-entity graphs.	Does not provide any comparison with state-of-the-art fraud detection methods.
(Smiles & Sasi Kumar, 2019)	Utilises SMOTE and ensemble sampling methods to improve machine learning performance	Identifies better algorithms in the methodology section before obtaining results.

Table 2-8 Critical Findings

SL	**Findings**	**Literature**
1	Lacks coherence and logical sequence between different sections of papers.	(Alarfaj et al., 2022) (Smiles and Sasi Kumar, 2019)
2	A lack of generalisation is observed. Use only one dataset and/or results are shown in significant differences in different datasets.	(Carrasco and Sicilia-Urbán, 2020) (Zhang and Huang, 2020)
3	Feature engineering is not addressed, and hyperparameters are used without optimisation.	(Dang et al., 2021) (Liu et al., 2020)
4	Lacks information on the practical implementation	(Li et al., 2020)
5	Performances are not compared with other state-of-art fraud detection methodologies. All measuring indexes such as Accuracy, Recall, Precision, F1, ROC, AUC, and Accuracy and Loss curve are not considered.	(Benchaji et al., 2021) (Kewei et al., 2021) (Naby et al., 2021) (Pratuzaite and Maknickiene, 2020) (Pumsirirat and Yan, 2018) (Shenvi et al., 2019) (Wang et al., 2021)
6	Lacks a summary of key findings. Do not provide analysis, discussion, or future implications of the project.	(Najadat et al., 2020) (Youssef et al., 2020)

2.6 Conclusion

This chapter has provided a systematic literature review on using DL models for fraudulent transaction detection to answer the research questions formulated in Chapter 1. Firstly, an appropriate methodology has been chosen to conduct the systematic literature review, including defining search strings, searching articles in scientific databases,

selecting articles based on predefined selection criteria, and data extraction, analysis, and synthesis. The answers to the research questions were derived through synthesis of the data.

For research question RQ1, Deep learning techniques for fraudulent transaction detection generally use the following six steps: 1. Data collection, 2. Data pre-processing, 3. Data pre-analysis, 4. Data splitting into training and testing sets, 5. Model training and optimisation, and 6. Model testing/evaluation. In most cases, electronic transaction datasets are imbalanced, so oversampling, undersampling or other techniques are used to mitigate the imbalance. For research question RQ2, the features are implicitly extracted from raw data by deep learning models such as CNN and autoencoder to train the classifier used for fraudulent transaction detection. CNN can generate features at different levels of abstraction, while the autoencoder can compress the feature representation through dimensionality reduction. For research question RQ3, the performance of CNN improves with the increasing number of layers until a certain point, after which the model tends to overfit. The exact number of layers needed depends on the nature of the data. Similarly, a higher number of filters makes more features available to train a classifier, resulting in a better performance. For research question RQ4, the performance of DL models varies across datasets, so that it cannot be said with certainty that a particular DL model outperforms others in all situations. The CNN and autoencoder models outperform classical and deep learning models for point data, while the RNN and LSTM models are widely used for sequence data. When autoencoder and CNN are used in tandem, where CNN uses the features extracted by autoencoder, it produces better results than passing raw data to CNN.

Lastly, the pros and cons and the critical findings from the literature are presented in a tabular format. Overall, the outcome of the review provides information on the use of DL models for fraudulent transaction detection as well as answers to the research questions.

The next chapter discusses the research methodology followed in this thesis, which includes research design and experiment design. The experiments are designed to validate hypotheses H1 and H2, formulated in Chapter 1.

Chapter 3. Research Methodology

3.1 Introduction

This chapter discusses the research methodology followed in this thesis. The previous chapter has derived answers to the research questions from a systematic literature review.

The findings from the literature review pertaining to RQ4, as illustrated in Section 2.4.1.4, suggest that deep learning (DL) methods such as Convolutional Neural Network (CNN) outperform classical machine learning algorithms in fraudulent transaction detection. CNN is useful for reducing the number of features in the input. It can extract important features from the input at different levels of abstraction (Ikeda et al., 2021). CNN has the property of translation invariance, which means that an object in an image or pattern in data can be recognised even when the pattern's position, orientation, or size varies (Biscione & Bowers, 2020). As fraudsters change their pattern of fraudulent activities, CNN is selected as a potential model in fraud detection.

The findings from the literature review pertaining to RQ2, as illustrated in Section 2.4.1.2, reveal that the relevant features from raw data are implicitly extracted by CNN without needing a separate feature extraction step. However, these features are not explainable, unlike the features generated by explicit feature engineering techniques. There are several explicit feature engineering techniques in the literature, such as Cognito (Khurana et al. (2016), Hidden Markov Model (HMM) (Lucas et al., 2020) and Deep Feature Synthesis (DFS) (Kanter & Veeramachaneni, 2015)). Of these, DFS has the unique capability to generate features from relational datasets linked by unique identifiers. It also can generate a rich feature space, and its performance is comparable to that of a human. Wedge et al. (2017) reported the enhanced performance of a random forest classifier when augmented with a DFS algorithm. Yang et al. (2021) also reported a performance enhancement achieved through a DFS algorithm in the context of car loan fraud detection. To the best of our knowledge, the impact of using DFS with CNN for fraudulent transaction detection is yet to be studied thoroughly.

A set of configurations needs to be set up for building a DL model. These configurations directly influence the characteristics and capability of the model. These configurations are referred to as hyperparameters, which are variables related to the model's structure and training process (Aszemi & Dominic, 2019). The findings from the literature review pertaining to RQ3, as illustrated in Section 2.4.1.3, demonstrate the importance of various hyperparameters in CNN, such as the number of hidden layers, filter size, number of epochs, batch size, learning rate, etc. To the best of our knowledge, the impact of hyperparameter optimisation on the fraudulent transaction detection performance of CNN is yet to be analysed thoroughly.

Section 3.2 defines the theory behind the research design and revisits the hypotheses constructed in Chapter 1. Section 3.3 provides details of the experiment design, which includes the following steps: data collection, data pre-processing, data pre-analysis, data-splitting, model training and optimisation, and model testing and evaluation. The experiment design outlined in this chapter is followed in Chapter 4 and Chapter 5 with slight variations.

3.2 Research Design

In this research, the scientific method is used to answer research questions. The scientific method is an empirical method involving observations, hypothesis formulation, conducting experiments, and analysing the results to validate the hypothesis (Honderich, 2005).

Based on the literature advocating the efficiency of CNN in fraud detection, as well as the performance enhancement attributed to DFS and hyperparameter optimisation, the hypotheses formulated in Chapter 1 are refined as follows:

> H1: DFS improves the fraudulent transaction detection performance of CNN significantly.
> H2: Hyperparameter optimisation significantly enhances the efficacy of CNN in detecting fraudulent transactions.

The process of experimenting and validating the above hypotheses is explained in the following section.

3.3 Experiment Design

The following steps have been carried out to prove the validity of the hypothesis mentioned in the earlier section. Electronic transaction data is collected to test the hypothesis, including both fraudulent and legitimate transaction data. The machine learning models are chosen in the first hypothesis. The data is then pre-processed, including oversampling of data belonging to the minority class and applying Deep Feature Synthesis (DFS) to extract features. As a preparation for training the chosen CNN model, the collected data is split into training and test datasets. The model is then trained with the training data, and model hyperparameters are optimised for the optimal performance. The performance evaluation of the chosen models is done using the test dataset. The whole process of the experiment is illustrated in Figure 3-1. A discussion of the steps involved in the experiment follows next.

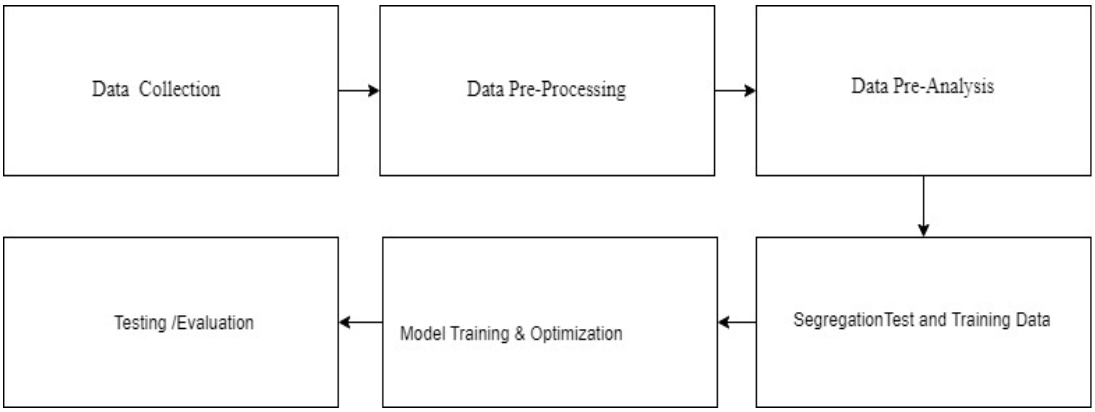


Figure 3-1 Steps of Fraud Detection Process.

3.3.1 Data Collection

Data is a crucial element of the machine learning method as well as any research field. Stojanović et al. (2021) explained that getting private data on financial fraud transactions is challenging due to reluctance to disclose this type of data, as it involves the risk of reputation loss or compromise of privacy. We collect the secondary data, which is publicly available for training and testing deep learning models. We collected data from https://www.kaggle.com/datasets/ealtman2019/credit-card-transactions provided by Erik Altman.

Due to privacy concerns, transaction with fraudulent data is scarce. The dataset generated by Erik R. Altman consists of realistic data for credit card transactions that mimic real transactions. The dataset is also labelled as fraudulent or legitimate. It is a combination of technical and domain knowledge. The domain knowledge encompasses the mechanisms of credit card transactions and customer spending behaviours.

In this data generation case, an individual customer approach is considered, where the created data is used to create a model for each individual customer. The broad characterisation of individual customer features includes occupation, age, income, geographic distribution, credit score, etc. The reason behind the selection of the synthetic data for other benefits:

a) Improved explainability - as the data is realistic in nature, it is helpful to explain the results.

b) Avoiding bias in results - the data is statistically checked to determine whether the two classes are identical.

c) Natural data is sparse or unbalanced- The natural data skews more towards the majority class in the anomaly detection problem, such as detecting fraudulent transactions.

The generated data are organised into three entities:

First, the customer information has been provided with the following features and data (in each line, the first part before the comma is represents features of dataset, and the part after the comma represents the data for the corresponding feature):

[Person],Hazel Robinson
[Current Age], 53
[Retirement Age], 66
[Birth Year], 1966
[Birth Month], 11
[Gender], Female
[Address], 462 Rose Lane
[Apartment], La Verne
[City], La Verne
[State], CA
[Zipcode], 91750
[Latitude], 34.15
[Longitude], -117.76
[Per Capita Income - Zipcode], $29278
[Yearly Income - Person], $59696
[Total Debt], $127613
[FICO Score], 787
[Num Credit Cards], 5

The user has 5 credit cards, and the information for one of the cards with feature names is:

[User],0
[CARD INDEX], 0
[Card Brand],Visa
[Card Type], Debit
[Card Number], 4344676511950444
[Expires], 12/2022
[CVV], 623
[Has Chip], YES
[Cards Issued], 2
[Credit Limit], $24295
[Acct Open Date], 09/2002
[Year PIN last Changed], 2008
[Card on Dark Web], No

The customer's card has thousands of transactions, and information for some of the cards is given below:

User,Card,Year,Month,Day,Time,Amount,Use Chip,Merchant Name,Merchant City,Merchant State,Zip,MCC,Errors?,Is Fraud?

0,0,2002,9,1,06:21,$134.09,SwipeTransaction,3527213246127876953,La Verne,CA,91750.0,5300,,No

0,0,2002,9,1,06:42,$38.48,Swipe Transaction,-727612092139916043,Monterey Park,CA,91754.0,5411,,No

0,0,2002,9,2,06:22,$120.34,Swipe Transaction,-727612092139916043,Monterey Park,CA,91754.0,5411,,No

0,0,2002,9,2,17:45,$128.95,Swipe Transaction,3414527459579106770,Monterey Park,CA,91754.0,5651,,No

0,0,2002,9,3,06:23,$104.71,Swipe Transaction,5817218446178736267,La Verne,CA,91750.0,5912,,No

0,0,2002,9,3,13:53,$86.19,Swipe Transaction,-7146670748125200898,Monterey Park,CA,91755.0,5970,,No

……. Many More transactions

3.3.2 Data Pre-processing

The data found in secondary sources are highly imbalanced. To balance the data, the majority class data needs to be undersampled, or the minority class data needs to be oversampled before the data is used. Oversampling techniques are used to create new samples of the minority class to create an equal number of samples for both classes. In our project, we use an undersampling method to balance the two classes in the dataset.

In the next step of data preprocessing, feature detection plays a vital role in data classification. In traditional machine learning models, domain experts identify or create features for training machine learning models. This process is known as ‘feature engineering’. However, deep learning models can automatically extract features at

various levels of representation, corresponding to different degrees of abstraction. Therefore, deep learning models do not require manual feature engineering and can capture complex nonlinear features.

We also use Deep Feature Synthesis (DFS) to extract features from the fraudulent transaction data. DFS is an automated feature engineering method which produces features that are explainable to humans, unlike deep learning techniques, which produce mostly unexplainable features. DPS has been shown to significantly reduce false positive rates (Wedge et al., 2017). The details of feature generation are further discussed in Chapter 4.

3.3.3 Data Pre-analysis

After pre-processing, the data is analysed using different statistical formulas: sum, mean, and standard deviation. This gives a primary idea about the data.

3.3.4 Splitting Data into the Training and Testing

The collected data is split into two sets, i.e., training and test datasets, with an 80: 20 ratio. That means 80% of the total data is used to train models, while the remaining 20% is used for testing models, and performance is measured using the test dataset.

3.3.5 Model Training and Optimisation

In this research, we choose a deep learning model, CNN, to detect fraudulent transactions. Literature suggests that these models can analyse large volumes of data and extract complex patterns from data (Galajit et al., 2019). CNN is configured by varying the hyperparameters, such as nodes, hidden layers and filters, by validating and testing model performance to optimise fraud detection accuracy (Becherer et al., 2019). The hyperparameters which provide the optimal results are selected for model configuration.

3.3.5.1 Convolutional Neural Networks

Convolution Neural Networks (CNN) is an upgraded version of Artificial Neural Networks (ANN), which can extract the feature itself through feature learning (O'Shea & Nash, 2015), and outperforms the traditional ANN because of capacity to automatically encode specific features into the architecture and reduce the number of parameters. CNN is constructed by three types of layers: convolutional, pooling and fully connected layers, and it is capable of extracting the spatial features automatically and adaptively using a

backpropagation technique (Yamashita et al., 2018). The architecture of CNN is illustrated in Figure 3-2.

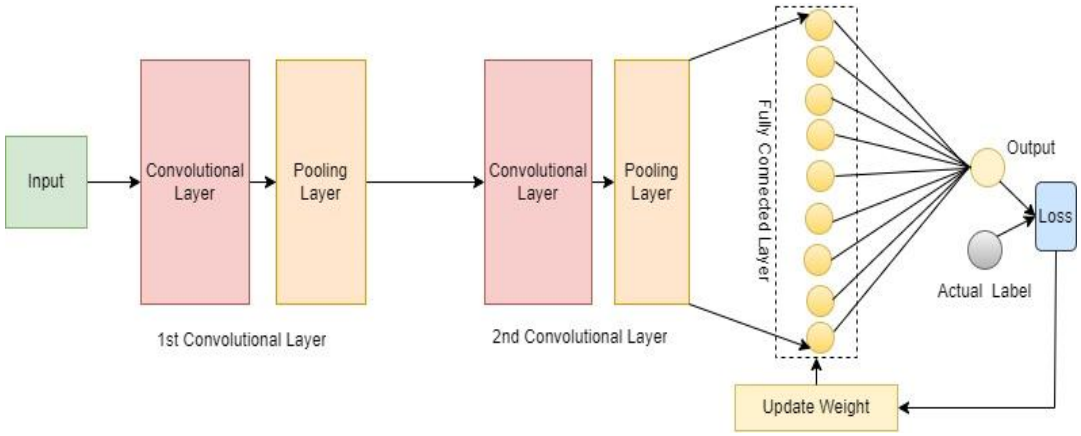


Figure 3-2 CNN Architecture.

The function of CNN can be divided into four key areas (Yamashita et al., 2018).

1. Like ANN, the first layer of CNN is the *input layer*, which receives the input data vector or matrix.

2. The second layer is the *convolutional layer*, which receives the output of the input layer by multiplication with weights. The convolutional layer is the central part of CNN that extracts features of a given input using linear and non-linear operations. A small array of numbers, called a kernel (or filter), is used in the convolutional operation. The kernel is applied across the input vector or matrix. An element-wise product between the input and the kernel is calculated and summed up to obtain an output called a feature map. The process is repeated for multiple kernels to form different featured maps representing various input characteristics. This layer uses a rectified linear unit (RLU) or sigmoid activation function (σ) to produce the output (Alzubaidi et al., 2021b).

3. The next layer is the *pooling layer*, which intakes the output of the convolutional layer. The pooling layer is used to downsample the spatial dimensionality of a given input and to reduce the number of parameters in the network. This layer helps reduce the computation complexity of CNN. The pooling layer is applied to each feature map independently by dividing it into non-overlapping regions and selecting a representative value for each region. Max pooling is the most used technique to retrieve the maximum value from the selected region. Other pooling techniques, such as average pooling and stochastic pooling, can also be used. The main tasks of pooling layers are:

a) *Spatial downsampling:* The pooling layer reduces the spatial resolution of feature maps by taking the maximum value or average value from each selected pooling region. This reduces the number of parameters and computational complexity in subsequent layers.

b) *Translation Invariance:* The transaction invariance property of CNN models enables them to capture the patterns regardless of their location. This means that a CNN can recognise a pattern or feature within a data sample or an image, regardless of where it appears spatially. Max pooling helps retain the salient features and ignores the less significant features in the feature map (Abdel-Hamid et al., 2012). This makes a network less sensitive to distortion or small translations in the input.

c) *Larger Receptive Field*: The receptor field is the input portion connected to the next layers using convolution or pooling operation. The area of input representing the reception field influences the computation of specific features. In the initial layer of a network, the reception field is small and limited to the local region. Convolutional and pooling layers are used to get a broader range of input in the deeper layer of a network. Pooling operation increases the receptive field of subsequent layers. However, it decreases the spatial dimension (Wu et al., 2020). Therefore, more significant features from input can be generated in the deeper layers of networks. The increased size of receptive fields plays an important role in more global and context-specific features from the input.

4. The *fully connected (or dense) layers* work like ANN and attempt to produce a class score from the activation and actual classification that happens in this layer. Fully connected layers appear at the end of a series of convolutional and pooling layers. The output of the final pooling layer is transferred as a one-dimensional vector to the first fully connected layers. This layer is connected fully to one or more fully connected layers before producing the final output. The purpose of this layer is to extract the complex patterns and high-level features to produce the final output for regression or classification. In these layers, an activation function such as Sigmoid, Rectified Linear Unit(ReLU), or SoftMax is used to build a non-linear relationship between input and output. The number of neurons is determined based on the specific task and the architecture of CNN. The number of neurons at the final output layer is the same as the number of classes in a classification problem. For example, to recognise five objects by a CNN, the number of

neurons at the output layer should be five. In the proposed project, we need to determine whether a transaction is fraud or not. Hence a CNN with a single neuron at the final output layer can classify fraudulent or legitimate transactions.

Training of CNN: Training of CNN is done by finding the optimum kernels in the convolutional layers and the optimum weights in the fully connected layers, which produces the minimum difference between the predicted classes at the output and the actual classes labelled in the training dataset. The kernels and weights in the convolutional and fully connected layers act as parameters of a CNN. The performance of CNN is calculated by a loss function where backpropagation and gradient descent are used for optimisation. Learnable weights are updated from the value generated from the loss function.

Loss Function: The loss function is an important hyperparameter of the CNN model. The loss function determines the deviation of the predicted output from the actual output. During the model training stage, weights are updated to minimise the loss function. Cross-entropy and mean squared error are used for the calculation of the deviation.

Gradient descent: Gradient descent is an iterative method to optimise an objective function using differentiable properties. It is used to find the minimum (or maximum) of the objective function, known as the "cost" or "loss" function. A random point is selected in the cost function to find the minimum point. Then the slope of the point is used to find the next lower point. The slope is considered as a gradient of the cost function. A small step is taken in the gradient's opposite direction to find the minimum (Baldi et al., 2000). The step size is called the learning rate. Mathematically, the weights (or other parameters) are updated by the practical derivate of the calculated cost or loss concerning the learnable parameters. The formula for a single update of weight is shown in equation 3.1:

$$\boldsymbol{w} := \boldsymbol{w} - \boldsymbol{\alpha} * \frac{\boldsymbol{\delta L}}{\boldsymbol{\delta w}} \qquad 3.1$$

where *w* denotes a weight in the network, α denotes the learning rate, and L denotes the loss function.

CNN is popularly used in different kinds of computer vision applications, as well as with text-based datasets, to efficiently perform classification tasks. The automatic

feature extraction capability of the CNN model makes it useful in the applications of object detection, image recognition, and self-driving cars (Bengio et al., 2017a).

3.3.5.2 Model Optimisation Techniques

Effective fraud detection relies not only on selecting appropriate models but also on optimising models to achieve optimal performance. To find the optimal performance, some hyperparameters are considered, such as the number of layers, filters, kernel size, pooling size, learning rate, activation function, dropout rate, and optimiser. Two optimisation algorithms – grid search and random search – are used to find the optimised model. The details of the optimisation are presented in Chapter 5.

3.3.6 Model Testing and Evaluation

Trained Models were tested with the test data (20% of data), which was separated in a previous stage and used to measure the performance of CNN. The results are recorded and evaluated in the evaluation stage.

3.3.6.1 Evaluation Metrics

There are various metrics for measuring the performance of machine learning models. We use several standard metrics to measure the detection accuracy of fraudulent and non-fraudulent transactions. Further, the efficiency metrics used by (Misra et al., 2020) are also used in this research to measure the efficiency of the chosen models, assuming the fraudulent class is the positive class. The performance of classification is measured using the following values: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN). The following formula is then used to determine the efficiency of the chosen models:

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+FP+TN+FN}} \quad 3.2$$

$$\boldsymbol{Precision(P)} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \quad 3.3$$

$$\boldsymbol{Recall\ (R)} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \quad 3.4$$

$$\boldsymbol{F1-Score} = \frac{\boldsymbol{2*P*R}}{\boldsymbol{P+R}} \quad 3.5$$

ROC and AUC curve:

Receiver Operating Characteristics (ROC) and Area Under Curve (AUC) are commonly used as evaluation metrics in machine learning, particularly in binary classification problems (Raghavan & Gayar, 2019). They help assess the performance of a classification model across different thresholds. The ROC curve is drawn using two indicators,True Positive Rate (TPR) and false Positive Rate (FPR), that can be defined as:

$$\boldsymbol{TPR} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \qquad 3.6$$

$$\boldsymbol{FPR} = \frac{\boldsymbol{FP}}{\boldsymbol{FP+TN}} \qquad 3.7$$

Here, TP, FP, TN, and FN denote the number of True Positives, False Positives, True Negatives, and False Negatives respectively.

The ROC curve is drawn by plotting TPR on the X-axis and FPR on the Y-axis. Figure 3-3 illustrates an ROC curve.

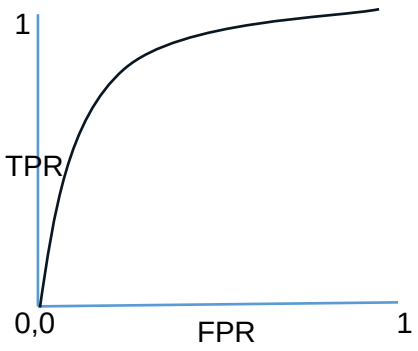


Figure 3-3 ROC curve.

The Area under the ROC curve (AUC) is calculated by measuring the two-dimensional area covered under the ROC curve from (0,0) to (1,0) (see Figure 3-4). The value of AUC ranges from 0 to 1. When a model's prediction is 100% inaccurate, the AUC value is 0. Conversely, when a model's prediction is 100% accurate, the AUC value is 1. An AUC of 0.5 indicates no discrimination, while an AUC above 0.7 is considered acceptable.

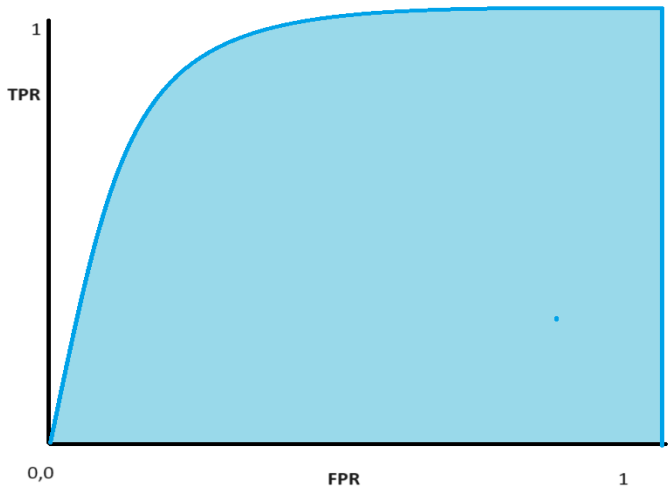


Figure 3-4 AUC (Area under ROC curve).

3.4 Summary

This chapter has provided a comprehensive discussion of the research methodology followed in this project. The chapter begins by discussing the research design, followed by the experiment design.

The research design has addressed the hypothesis to be tested via the experiment. The experiment design includes data collection, data pre-processing, data pre-analysis, data segregation, model training and optimisation, and evaluation. Moreover, this chapter has explored the evaluation criteria for assessing the performance of the DL models. The experiment design outlined in this chapter is followed in Chapter 4 and Chapter 5, with slight variations, depending on the use of feature engineering, model training and optimisation methods. This chapter has also presented an elaborate discussion of the CNN architecture and the training process to set the background for the subsequent chapters.

In the following chapter, we focus on feature engineering techniques. Feature engineering can play a crucial role in extracting the critical features from the data to improve the efficiency of the CNN model for fraudulent transaction detection.

Chapter 4. Feature Engineering for Fraud Detection

4.1 Introduction

This chapter validates the hypothesis H1 formulated in Chapter 1 and refined in Chapter 3. The chapter first generates a set of interpretable features from an existing fraudulent transaction dataset using Deep Feature Synthesis (DFS), and uses those features to train a Convolutional Neural Network (CNN) model. The effects of using DFS on the performance of the CNN model in detecting fraudulent transactions are evaluated to validate the hypothesis.

In the classification problem in machine learning, selecting the right features is crucial. It is useful to create new features from the existing dataset to improve the predictive performance of machine learning. Feature engineering involves applying mathematical operations such as arithmetic and aggregate functions to existing features to generate new ones (Nargesian et al., 2017). Manual and automatic feature engineering have been widely used for feature generation in the literature (Chang et al., 2022). Unlike manual feature engineering, automated feature engineering is efficient due to being less labour-intensive and less dependent on human domain expertise. While deep learning models can extract features from the dataset without requiring an explicit feature extraction step, it is often challenging to interpret them, as most deep learning models operate as black boxes. A drawback of using the implicit feature engineering of deep learning in fraudulent transaction detection is that it initially performs effectively, but that performance degrades with time due to the dynamic nature of fraudulent transactions (Chang et al., 2022). On the other hand, explicit feature engineering has the potential to extract explainable features, and it is independent of the model being used, which makes it usable with any model.

Many explicit feature engineering methods have been used to generate features for machine learning. Some of the explicit feature engineering techniques are Cognito, the Hidden Markov Model (HMM) and Deep Feature Synthesis (DFS). Cognito, an automated feature engineering technique for supervised learning proposed by Khurana et al. (2016), generates feature from existing feature and makes a hierarchy or tree for accuracy with the different combinations of the generated feature. Features are selected

from the tree for the accuracy maximisation through a greedy search algorithm. The Hidden Markov Model (HMM) (Lucas et al., 2020) is used to predict the likelihood of the following transaction from previous sequences of transactions. HMM considers the transaction as a sequence of events rather than an isolated event and models the sequence from different perspectives. A classifier uses Likelihoods as an additional feature in fraud detection techniques. This approach increases the effectiveness of classifying fraud transaction detection when combined with other expert-based fraudulent transaction detection systems (Lucas et al., 2020). Finally, the Deep Feature Synthesis (DFS) algorithm proposed by Kanter and Veeramachaneni (2015) can generate features from relational datasets linked by unique identifiers; it can generate rich feature space, and its performance is matched with a human level at the data science competition (Kanter & Veeramachaneni, 2015). This feature engineering technique automatically generates features from existing features by applying mathematical functions based on the relationship among the datasets. DFS-extracted features are used to improve the performance of several traditional machine learning models. However, to the best of our knowledge, the impact of DFS on deep learning has yet to be studied thoroughly.

In the experiment, we first measure the performance of the CNN deep learning model for fraudulent transaction detection using a dataset found in an open-source platform. In this case, the CNN classifier only utilises the implicit features extracted from the raw data by the early layers of the CNN model. Then features are explicitly generated from the dataset using the DFS algorithm, after the preprocessing stage, to be fed into the CNN model with the same parameters and hyperparameters. In this case, the CNN classifier utilises the explicit features generated from the raw data by the DFS algorithm. The early layers of CNN extract more abstract features from those explicit features. Finally, we measure and compare the performance indexes in both cases.

The following section presents relevant works on using DFS with machine learning models. Section 4.3 describes the experiment setup with an elaboration of the DFS technique. Section 4.4 discusses the dataset and experimental results, and Section 4.5 concludes this chapter by summarising the findings.

4.2 Related Works

Hu et al. (2019) proposed a feature engineering method, namely Midway Neural Networks, using deep neural networks for data mining from high-dimensional event logs. The method eliminated the manual feature engineering and repeated pre-processing steps of event logs. Their proposed method takes the input in a particular time window in a dense representation of networks for incremental training and prediction.

Wang et al. (2020) proposed a label-based Regression method using correlation between the pair of features. Their method captures the pair of features with higher discrimination, and this method improves the accuracy by 3.13% compared to the original feature.

Deep Feature Synthesis (DFS) has been used with many traditional machine learning classifiers. Wedge et al. (2017) used the DFS algorithm and a random forest classifier, automatically extracting features from a large volume of data and significantly reducing the false positive rate. The authors claimed that their approach, which was applied to 1.852 million transactions, reduced the false-positive alerts by 54%. Yang et al. (2021) investigated the performance improvement in using DFS for car loan fraud detection and found that the accuracy improved by 23%. Chang et al. (2022) used DFS to predict fraud in online credit loan applications before loan approval, where the data was captured from loan forms and then imported into a graph database with graph features. The basic features from the graph database were split into several relational tables. Using DFS, new synthetic features were generated automatically from the tables. Finally, a fraud risk assessment model was built based on those features. Their model's efficiency was higher than those constructed manually by business experts.

Besides financial fraud detection, DFS has been used with use cases related to the education, medical and engineering sectors. DFS feature engineering was used with the K-nearest neighbours classifier by Mohamad et al. (2020) to predict student performance in a massive open online course, which had a prediction accuracy of about 80%. Gür (2022) used DFS with machine learning techniques to predict breast cancer, where performance increased when DFS was used with the ML models, and the DFS + Logistic Regression model performed best. DFS was used by Rao and Li (2021) to predict the

remaining useful lifetime of a motor. After collecting data from sensors during the motor life cycle, features were generated from the collected raw data using DFS. The generated features were then used to train an LSTM model for time series forecasting.

4.3 Experiment Setup

To explore the impact of feature engineering on the deep learning model, our experiment follows some sequential steps (see Figure 4-1), which are as follows: data collection (A), data preprocessing (B), feature engineering (C), classification (D), and performance measurement (E).

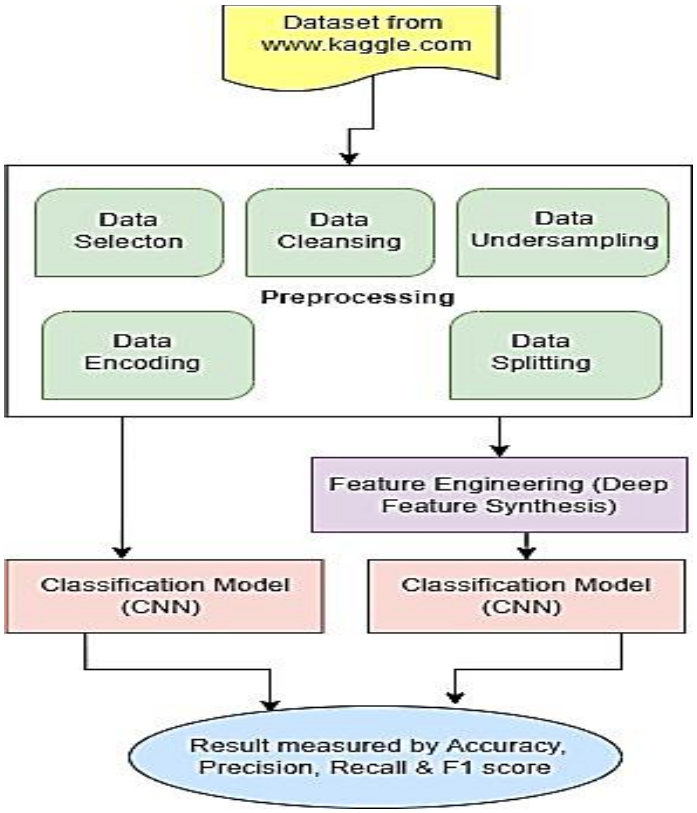


Figure 4-1 The Steps Involved in the Experiment Design.

The experiment is conducted for two separate cases. In Case 1, steps A, B, D, and E were followed, and in Case 2, the steps A, B, C, D, and E were followed. The details of each step are described in the following sub-section.

A) Data Collection

The dataset for the experiment is collected from the open-source platform (https://www.kaggle.com/datasets/ealtman2019/credit-card-transactions). The experiment used two tables from the dataset, *credit_card_ transactions-ibm_v2.csv (transaction file)*

and *sd254_cards.csv (card file)*, consisting of 15 and 13 fields, respectively.

B) Preprocessing

The data preprocessing stage of the experiment comprises the following steps:

- *Data selection:* Deep feature synthesis creates features from relational database tables. To correlate data across tables, data is grouped by payment cards from *sd254cards.csv*, and the corresponding transactions from *credit_card_transactions-ibm_v2.csv* pertaining to that user are selected.
- *Data Cleansing:* null data are replaced with zeros, and invalid characters, such as the dollar sign beside the transaction amount, are removed at this stage.
- *Data Undersampling:* The dataset was found to be highly imbalanced, where the ratio of legitimate transaction records (2,452,188) to fraudulent transaction records (1,432) in the dataset is 99.94:0.058. To balance the majority class with the minority one, an undersampling technique is used. In the undersampling method, the number of records randomly selected from the majority class equals the number of records originally in the minority class, which equalises the majority and monitory class sample sizes.
- *Data Encoding:* Categorical features or text type fields are transformed into numerical values using Label Encoder.
- *Data Splitting:* The dataset is split into training and test datasets with an 80:20 ratio. 20% of the training dataset is used as a validation dataset.

C) Feature Engineering (using Deep Feature Synthesis)

In this experiment, DFS, proposed by Kanter and Veeramachaneni (2015), is used to create features for the CNN deep learning model.

Before going through the deep insight into DFS, let us introduce the basic terminologies used here.

Entity: Entity is defined by Chen (1976) as “An entity is a “thing” which can be distinctly identified.”

Person, Institution, Place, Thing, and Event are examples of the entity. In a relational database, entity information is stored in a single and multiple related tables.

Feature: The attributes or characteristics that belong to an entity are known as features. For example, Name and Date of Birth are the features of a Person entity. Features of an entity are stored as columns in a relational database table.

Instance: An instance refers to a single, specific occurrence or observation of an entity at a particular point in time. It represents a unique record in a dataset. For example, each occurrence of all attributes of a person entity for different individuals is the instance of the entity. In a database table, each row is an instance of the entity.

In mathematical notation, entities in a dataset are expressed as $E^{1,\dots,K}$ for k number of entities, each entity having i instances.

Here, a specific feature of an entity is denoted as $x_{i,j}^{k}$.

In this context, the above notation represents the j^{th} feature of the i^{th} instance of the k^{th} entity. We can conceptualise the data stored in a table as follows: the entity information is stored in tables, where the features of entities are logically linked to the columns of the tables, and each row corresponds to an instance of the entity.

Forward relationship (E_F): This is the relationship between a single instance of entity E^l with a single instance of another entity E^k.

Backward relationship (E_B): This is the relationship between a single instance of E^l to a collection or many instances of another entity E^k.

DFS is used to generate features automatically from multi-table transactional datasets. DFS generates features based on three types of functions. Firstly, an *entity feature* (EFEAT) for a particular candidate entity feature is created by applying a function to the entire set of values for that feature. Feature generated by an EFEAT function for the j^{th} feature for all instances can be expressed as

$$\boldsymbol{x_{i,j'}^{k}} = \boldsymbol{EFEAT}\left(\boldsymbol{x_{:j}^{k}}\right) \qquad 4.1$$

Here, the new entity feature $x_{i,j'}$ is generated by applying the EFEAT () function on the j^{th} column of all rows (instances). For example, from the "Date of Birth" feature in the Person entity, the new feature "Current Age" is generated by applying an EFEAT() function on the "Date of Birth" feature.

Secondly, a *direct feature* (DFEAT) is derived from two candidate entities (E^l, E^k) that are linked via a Forward Relationship (E_F), i.e., a single instance of E^l is directly mapped to a single instance of E^k. The feature values of E^l are directly transferred as new feature values of E^k:

$$\boldsymbol{x}_{i,j'}^{\boldsymbol{k}} = \mathbf{DFEAT}(\mathbf{x}^{\mathbf{l}}) \quad 4.2$$

Thirdly, a *relational feature* (RFEAT) deals with a Backward Relationship (E_B), where one instance of a candidate entity (E^k) is linked to many instances of another candidate entity (E^l). A new feature value for entity E^k is generated from entity E^l by applying an RFEAT function on the corresponding values of the feature of E^l:

$$\text{The generated feature, } \boldsymbol{x}_{i,j'}^{\boldsymbol{k}} = \mathbf{RFEAT}(\mathbf{x}_{:\mathbf{j}/\mathbf{e}^{\mathbf{k}}=\mathbf{i}}^{\mathbf{l}}) \quad 4.3$$

Here, the new feature is extracted for entity E^k by applying the RFEAT function to $x^l_{:j/e^k=i}$, that is, all values of the *j*th feature of entity E^l in which the key instance is equal to the ith instance of Entity E^k.

The pseudocode of the function F^i to generate features for the i^{th} entity is provided in Table 4-1.

Table 4-1 Algorithm for Generating Features for the Target Entity.

DFS Algorithm (Wedge et al., 2019) :

function MAKE_FEATURES(E^i, $E^{1:M}$, E_V)
$E_V = E_V \cup E^i$
E_B = BACKWARD(E^i, $E^{1\ldots M}$)
E_F = FORWARD(E^i, $E^{1\ldots M}$)
for $E^j \in E_B$ **do**
MAKE_FEATURES (E^j, $E^{1\ldots M}$, E_V)
$F^j = F^j \cup$ RFEAT(E^i, E^j)
for $E^j \in E_F$ **do**
If $E^j \in E_V$ then
CONTINUE
MAKE_FEATURES(E^j, $E^{1\ldots M}$, E_V)
$F^i = F^i \cup$ DEFEAT(E^i, E^j)
$F^i = F^i \cup$ EFEAT (E^i)

How to generate features using DFS is discussed below:

Assume a dataset of *K* entities, i.e., $E^{1,}$..., E^K.

The DFS extracts features for entity E^k from all entities in the dataset, using RFEAT, DFEAT, and EFEAT functions, provided that other entities have Forward (E_F) and Backward (E_B) relationships with E^k.

As features generated by the EFEAT function stays in entity E^k, the feature created using the RFEAT and DFEAT functions are added with features created by EFEAT.

To generate RFEAT features for E^k features from entities in the E_B relationship

are used. All of the features for each entity in E_B are generated before the realisation of RFEAT features for E^K. Similarly, the DFEAT features for E^k are generated from entities in the E_F relationship. So all features of each entity in E_F are calculated first. Finally, RFEAT and EFEAT features are added to E^k.

This algorithm creates a pseudocode of function MAKE_FEATURES to generate features (F_i) for the E^i Entity. The method of organising recursive calls is outlined in the pseudocode. The calculation for each feature type follows the approach discussed above, i.e., RFEAT, DFEAT, and EFEAT generate features according to their respective types based on the input provided to the function. E_v is responsible for keeping track of visited entities to ensure that none of them are included again. $E^{1...M}$ denotes the M number of entities in the dataset, and "**for** $E^j \in E_B$ **do** " means that a repeated task for each entity in a backward relationship in the entity set E_B. "**for** $E^j \in E_F$ **do**" also means a loop operation for the repeated tasks for each entity in a forward relationship in the entity set E_F.

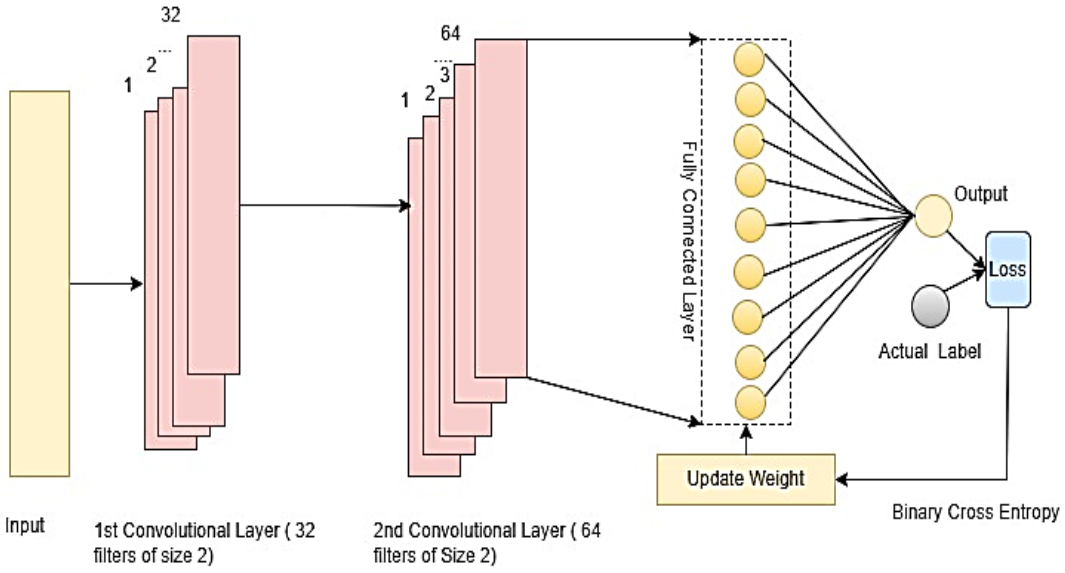


Figure 4-2 1d CNN Model with Two Convolutional Layers.

D) CNN Model Training

The problem addressed in this chapter is a binary classification problem. In the experiment, a 1-dimensional CNN model is used as a classifier to detect fraudulent transactions in both explicit and implicit feature extraction cases. In the latter case, the

CNN model is trained using raw data, while, in the former case, the CNN model is trained with features extracted with DFS. The model is set to undergo 20 training epochs with a learning rate 0.0001, using the provided training and test data. The input data is expected to have a shape defined by the number of data points and a single feature per point. The architecture involves two 1D convolutional layers: the first with 32 filters of size 2, activated by Rectified Linear Units (ReLU), and the second with 64 filters of the same size and activation function (Refer to Figure 4-2). Following these convolutional layers, the output is flattened for integration into a fully connected layer with 64 neurons activated by ReLU. A dropout layer with a rate of 0.50 is applied for regularisation, mitigating overfitting. Finally, a single neuron with a sigmoid activation function is employed for the binary classification task, and the model is compiled with binary cross-entropy as the loss function, the Adam optimiser, and the specified learning rate. The pseudocode for the model construction is described in Table 4-2.

Table 4-2 Pseudocode of CNN Model

```
1 : CNN(trainX, trainY, testX, testY, learningrate = 0.0001, epoch = 20)
2 : inputs = shape(datapoints, 1)
3 : model ← Conv1D( filters = 32(2),activation = ReLU)(input)
4 : model ← Conv1D( filters = 64(2), activation = ReLU)(model)
5: model ← Flatten()(model)
6: model ← Dense(neurons = 64, activation = ReLU)(model)
7 : model ← Dropout(0.50)(model)
8 : model ← Dense(neurons = 1, activation = sigmoid)(model)
9 : model.compile(loss = BinaryCrossEntropy, optimizer = Adam, learningrate)
```

E) CNN Model Testing and Evaluation

The performance of the standalone CNN model is evaluated with raw training data as input, while the performance of the CNN model with DFS is evaluated with features extracted explicitly by DFS as input. A confusion matrix is formed with True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) values for each case to measure the performance improvement due to using DFS.

The Accuracy (a), Precision (b), Recall (c), and F1 (d) scores for each case are calculated according to the formula below:

a) Accuracy= $\frac{TP+TN}{TP+FP+TN+FN}$

b) $Precision(P) = \frac{TP}{TP+FP}$

c) $Recall(R) = \frac{TP}{TP+FN}$

d) $F1 - Score = \frac{2*P*R}{P+R}$

In addition, ROC (Receiver Operating Characteristics) and AUC (Area Under the Curve) values are calculated to measure the performance for both cases (Hanley & McNeil, 1982).

4.4 Experimental Results

The unique instances of cards are identified from the instances of the card data table mentioned in Subsection 4.3.A. As a result, 41,615 such cards are found in this table. Afterwards, 2,452,188 records from the transaction data table are matched against the unique card instances. The statistics of the data records are given in.

Table 4-3 Dataset Information

Table column subhead	***Card (Record, Column Number)***	***Transaction (Record, Column Number)***
Primarily selected data	41,615	245,218,813
Fraud transactional data	-	1,432
Legitimate transactional data	-	2,452,188

The dataset contains a significantly higher number of legitimate transaction records than fraudulent ones. To resolve this imbalance, we randomly selected the same number of transaction records from the majority class as there were in the minority class. This results in an equal number of fraudulent and legitimate transaction records, totalling 1,432 records.

Case 1: Standalone CNN model

In Experiment Case 1, the standalone CNN model described in Figure 4.2 is used to classify legitimate and fraudulent transactions. The accuracy vs. the number of epochs curves are illustrated in Figure 4-3. The accuracy steadily increases for the training and validation datasets as the number of epochs increases. However, the validation accuracy

is always lower than the training accuracy. So, the model is overfitted when a standalone CNN model is used.

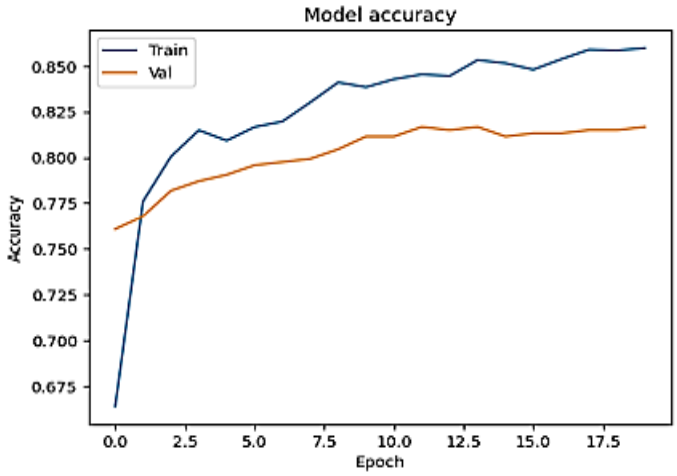


Figure 4-3 Accuracy vs. Number of Epochs Curves for the Standalone CNN Model

The model loss vs. the number of epochs curves is illustrated in Figure 4-4. Initially, the model loss for the training set is higher than that for the validation set. However, after epoch number 2, the model loss for the training set comes lower than the validation set up to epoch number 20.

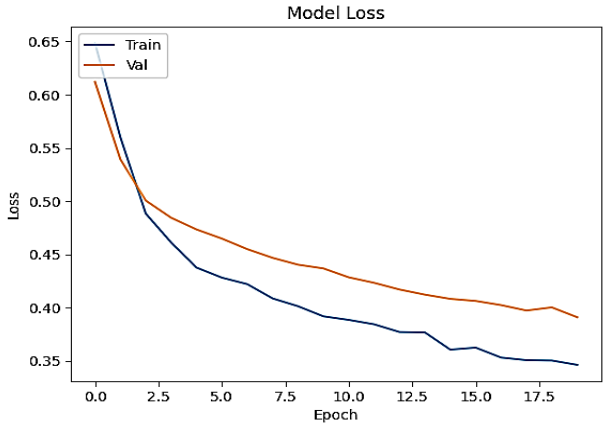


Figure 4-4 Model Loss vs. Number of Epochs Curves for the Standalone CNN Model.

The ROC curve for the standalone CNN model is provided in

Figure 4-5, where the AUC (area under ROC) was 0.90.

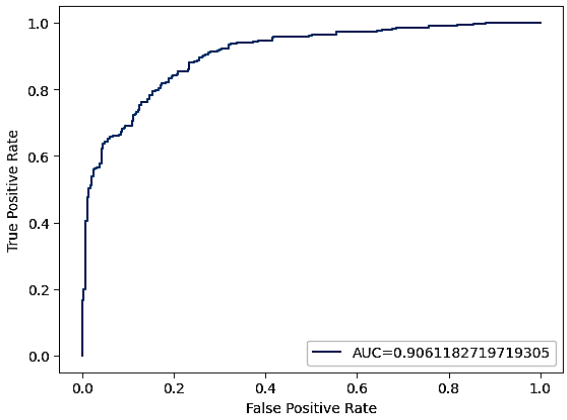


Figure 4-5 ROC Curve for the Standalone CNN Model.

Case 2: CNN model with DFS

In Experiment Case 2, the DFS feature engineering technique generates explicit features automatically from the card and transaction datasets, which are then fed into the same CNN model without changing the parameters or hyperparameters to classify legitimate and fraudulent transactions. Using 15 columns from the card dataset and 10 columns from the transaction dataset, a total of 119 columns are generated (see Appendix A). The accuracy vs. number of epochs curves are presented in Figure 4-6, where the accuracy for both the training and validation datasets increases as the number of epochs increases. The validation accuracy is still lower than the training accuracy. However, the difference between these two curves is more minor compared to the standalone CNN model. The accuracy of the training dataset improves from Case 1 (Figure 4-3) to Case 2 (Figure 4-6) due to using DFS feature engineering in the latter case. A similar trend is also observed for the validation dataset.

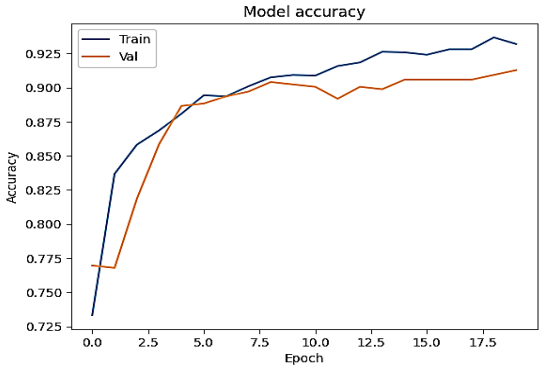


Figure 4-6 Accuracy vs. Number of Epoch Curves for the CNN Model with DFS.

The model loss vs. the number of epochs curves is presented in Figure 4-7, where the model loss for both the training and validation datasets decreases as the number of epochs increases. The model loss for the training set is higher than that for the validation set. However, the difference between the training and validation loss curves increases after 7.5 epochs. The model loss for the training and validation sets is lower for the CNN model with DFS than for the standalone CNN model.

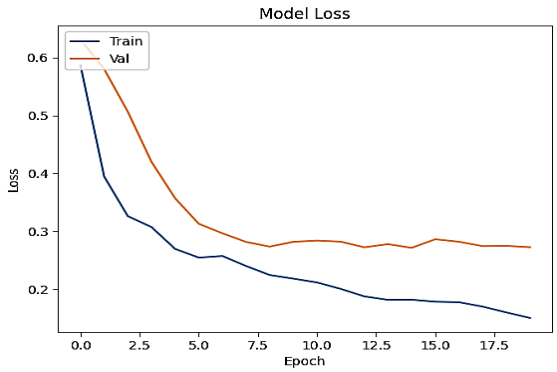


Figure 4-7 Model Loss vs. Number of Epochs Curves for the CNN Model with DFS.

The ROC curve for the CNN model with DFS is presented in Figure 4-8. The AUC increases by 7% from 0.90 in Case 1 to 0.96 in Case 2, attributed to the use of the DFS feature engineering technique.

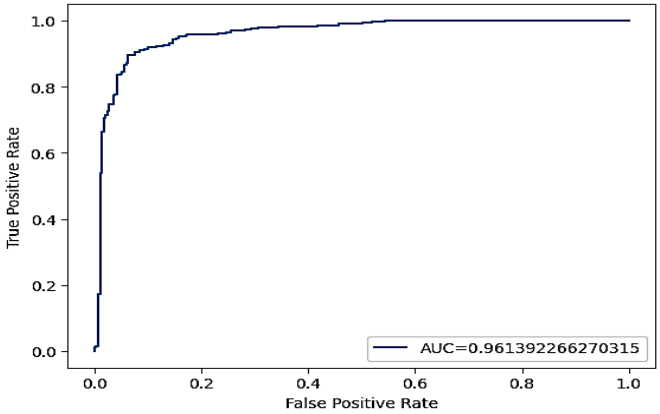


Figure 4-8 ROC Curve for CNN Model with DFS.

The performance indicators, such as Precision, Recall, F1 score, and Accuracy for both Case 1 and Case 2, are provided (Table 4-4). The precision for fraudulent transaction classification improves from 0.82 for the standalone CNN model to 0.92 for the CNN model with DFS, which is an improvement of 12% after applying DFS. Similarly, there are improvements observed for recall (0.80 to 0.90, an improvement of 13%), F1-score (0.82 to 0.91, an improvement of 11%), and accuracy (0.82 to 0.91, an improvement of 11%).

Moreover, the precision for legitimate transaction classification also increases from 0.81 for the standalone CNN model to 0.90 for the CNN model with DFS, which is an improvement of 11%. Similarly, there are improvements observed for recall (0.83 to 0.91, an improvement of 8%), F1-score (0.82 to 0.91, an improvement of 11%), and accuracy (0.82 to 0.91, an improvement of 11%).

Table 4-4 Performance Data for Case 1 and Case 2

Class	**Standalone CNN (Case 1)**				**CNN with DFS (Case 2)**			
	Precision	***Recall***	***F1-Score***	***Accuracy***	***Precision***	***Recall***	***F1-Score***	***Accuracy***
Fraud	0.82	0.80	0.81	0.82	0.92	0.90	0.91	0.91
Legitimate	0.81	0.83	0.82	0.82	0.90	0.92	0.91	0.91
Average	0.82	0.82	0.82	0.82	0.91	0.91	0.91	0.91

Figure 4-9 shows the average value of all performance indices across fraudulent and legitimate transactions. Average precision increased from 0.82 to 0.90 (10%), and, recall,

F1 Score, and accuracy increased by 11% (.82 to 0.91) from the standalone CNN model to the CNN model with DFS.

The above comparative analysis shows that, even though CNN has an inherent capacity to extract features from a dataset, using an explicit feature engineering step can significantly boost the performance of CNN. Like CNN's implicit feature extraction process, DFS can also extract features automatically without much human intervention.

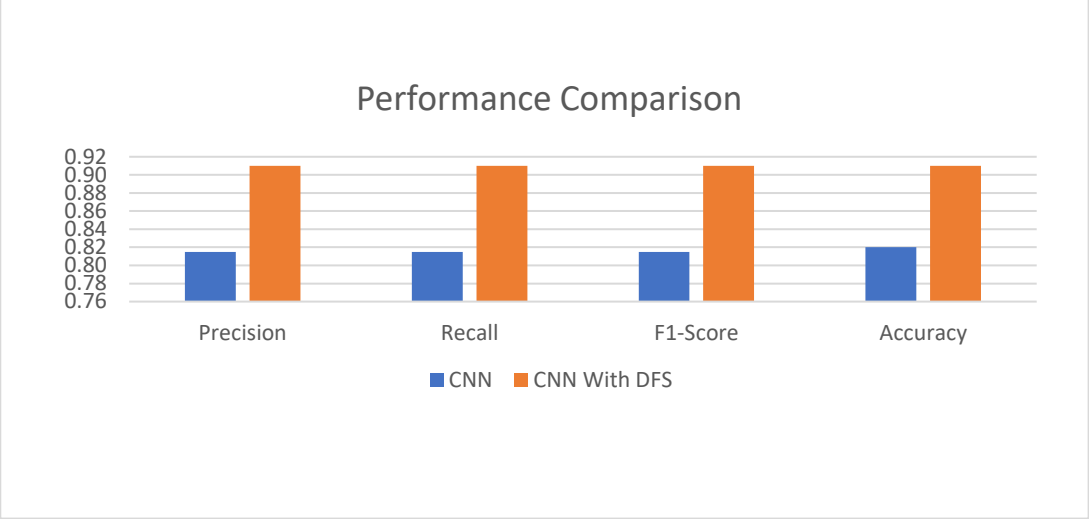


Figure 4-9 Bar Graph for the Average Performance Indexes of Fraud Detection for the Standalone CNN Model and the CNN Model with DFS.

4.5 Conclusion

This chapter explores the use of DFS as an automated technique for feature engineering to improve the performance of a 1-dimensional CNN model in detecting fraudulent transactions. To measure the impact of DFS on performance, we have compared the standalone CNN model's performance with that of the same CNN model with DFS, using the same dataset for both models. The experimental results have shown that the CNN model employing DFS achieves a notable 11% improvement in recall rate, F1 score, and accuracy, with a 10% improvement in precision. Therefore, hypothesis H1 is validated.

DFS has the added advantage of extracting features from raw data without manual intervention. Although CNN can also implicitly extract features, the features generated by it are usually incomprehensible to humans. In contrast, the features produced by DFS are

easily interpretable. In this experiment, an undersampling method has been utilised to tackle the data imbalance issue.

In the following chapter, we employ hyperparameter optimisation algorithms to maximise the performance of a CNN model in detecting fraudulent transactions. Appropriate selection of hyperparameters is vital to enable a DL model to capture the best features from the data to exhibit the best performance.

Chapter 5. Optimisation of CNN Model

5.1 Introduction

This chapter validates hypothesis H2 formulated in Chapter 1 and refined in Chapter 3. In this chapter, we identify the hyperparameters that result in the best performance for CNN in fraudulent transaction detection using hyperparameter optimisation algorithms. In the experiment, a baseline CNN model with default hyperparameter settings is first built and trained using training data. Later, we search for the optimal hyperparameter setting using hyperparameter optimisation algorithms. The optimisation algorithms use trial and error by building a CNN model with each new setting to see which gives the best performance. The performance of the optimised CNN models is compared against the baseline CNN model with default setting as well as a state-of-the-art model to validate the hypothesis.

When building a deep learning (DL) model, a set of configurations is required, not directly learned from the training data, which are adjusted by the machine learning researcher or an automated process. These configurations are referred to as hyperparameters, and they encompass the variables related to the network's structure and training process (Aszemi & Dominic, 2019). Hyperparameters are critical components of a DL model that significantly impact the training process as well as the model's performance.

Convolutional neural networks (CNNs) consist of multiple parameters and hyperparameters, which significantly affect their performance. The parameters, such as the weights and biases of CNN, are determined by the backpropagation algorithm during the learning phase (Lydia & Francis, 2019), so the parameters are directly learned from the training data. On the other hand, hyperparameters are selected by the machine learning researcher or an automated process, so these are not directly influenced by training data values. However, hyperparameters play a crucial role in the training process of the CNN model, influencing the parameter (i.e., weights and biases of CNN) updates during the learning phase. To find the hyperparameters that yield the most efficient model for fraudulent transaction detection, evaluating the model's performance across various values for each hyperparameter is necessary. This exploration of different hyperparameter values is often referred to as the hyperparameter search space. A broader search space for

each hyperparameter offers a more comprehensive range of possibilities for finding the optimal model configuration and training process. Searching for optimal performance within an extensive hyperparameter space demands additional resources and time. Finding the optimal composition and selection of hyperparameters is challenging. Several algorithms are employed to fine-tune convolutional neural network models, which this chapter discusses. Among those, two algorithms – random search (Bergstra & Bengio, 2012) and grid search (Liashchynskyi & Liashchynskyi, 2019) – are used to choose the best hyperparameters. In the case of CNNs, the weights and biases are the parameters that need to be updated based on the model's objective. On the other hand, the elements that affect the parameter updates are known as hyperparameters. For CNNs, hyperparameters include the size of filters, the number of filters, the size of convolutional layers, the number of layers, activation functions, optimiser selection, and learning rate.

The following section discusses the hyperparameters used in CNN models. Section 5.3 highlights the importance of hyperparameter optimisation, while Section 5.4 illustrates the hyperparameter optimisation algorithms used in this chapter. Section 5.5 presents the experiment setup, and Section 5.6 discusses the experimental results. Section 5.7 concludes this chapter by summarising the findings.

5.2 Hyperparameters in CNNs

The hyperparameters were discussed briefly in Chapter 3. In this chapter, each of the hyperparameters is discussed in detail.

Number of Layers: A CNN model comprises input, convolutional, pooling, and fully connected (also known as dense) layers.

In the convolutional layer, one or multiple filters are applied to the input features, performing a convolutional mathematical operation that generates a feature map for each filter. These convolutional layers are responsible for extracting abstract features from the input. The model can include multiple convolutional layers, each contributing to the extraction of different levels of abstraction. However, while more features can be extracted through additional layers, an excessive number of layers can also make the model excessively complex to learn and sometimes lead to overfitting. Hence, it is crucial to carefully determine the number of layers to optimise the model's performance. A very

typical configuration of a 1D CNN model is given in Figure 5-1. The model consists of the input, convolutional, pooling, flattened, fully connected, and output layers.

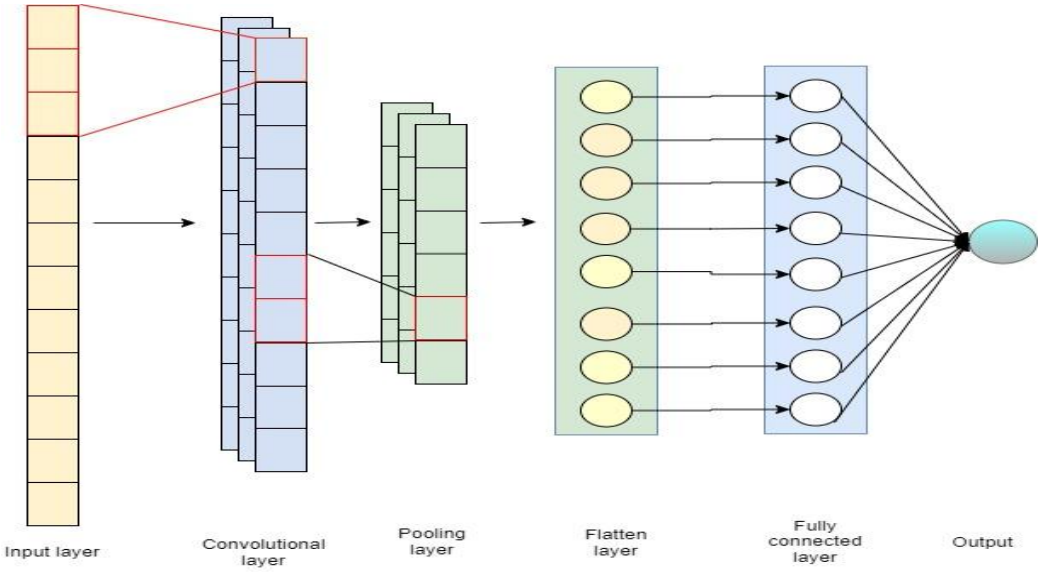


Figure 5-1 A Typical Configuration of a 1D CNN Model.

Filter/ Kernel Size: Filters, known as kernel masks or convolution matrices, perform operations like blurring, shaping, amplification, and feature extraction on input data or images (Bengio et al., 2017a). The use of odd-numbered filter sizes (such as 3, 5, 7, etc.) is prevalent due to the presence of a central point. This main point ensures an equal number of pixels or sizes on all sides of the centre, which aids in maintaining spatial symmetry during the convolution process. In Figure 5-1, a filter of size 3 is used to generate feature maps in the convolution layers.

Pooling Size: The pooling layer is utilised after the convolutional layer to decrease the dimension of feature maps through various matrix operations. Commonly, max pooling and average pooling are employed. This layer extracts significant features from the feature maps generated by the preceding convolutional layer. The pooling size, as well as the number of pooling layers, influences CNN's training time through the reduction of the number of parameters. A pooling size of 2 is used in Figure 5-1.

Learning Rate: During the model's learning process, parameters are initially assigned, which are then updated using the gradient descent algorithm. This algorithm updates the parameters gradually, with each step involving a minor adjustment. This step size is referred to as the learning rate (Alzubaidi et al., 2021a). In the process of updating

the parameter, especially in supervised learning scenarios, the learning rate plays a crucial role.

Activation Function: After being multiplied by weights and added by biases, the input of a fully connected layer is transformed by a function known as an activation function. These functions play a crucial role in classification and regression tasks in machine learning due to their ability to introduce non-linearity. Among various activation functions, the rectified linear unit (ReLU) and hyperbolic tangent (tanh) functions are widely used and popular choices (Alzubaidi et al., 2021a).

Dropout Rate: Dropout is a regularisation technique employed to address the issue of overfitting in a model. When a model is prone to overfitting, dropping out, or deactivating, a portion of neurons helps mitigate this problem. Applying dropout with varying percentages of neuron deactivation enhances a model's generalisation capability (Mohammed et al., 2022).

Batch Size: The batch size refers to the number of input samples fed into the model during training in a single iteration (Kim et al., 2020). All samples in a training set are processed in a single epoch. For instance, if a dataset contains 1000 records and the chosen batch size is 50, then the number of iterations in one epoch would be the total dataset size divided by the batch size, which is 20 in this case.

Mathematically, *the number of iterations per epoch = total dataset size/batch size*.

Larger batch sizes generally lead to shorter training times. However, larger batch sizes demand greater computational resources. Interestingly, even if a machine can accommodate a larger batch size, the model's performance might deteriorate, resulting in poorer generalisation capabilities. In such cases, determining the appropriate batch size involves fine-tuning the model using the dataset.

Optimiser: The weight update process of CNN is dictated by a loss function, which measures the difference between the actual (correct) values and the predicted values produced by the network. The optimiser is crucial in helping the model learn efficiently and strive for perfection during training (Alzubaidi et al., 2021a). There are different types of optimisers, each with its approach to guiding the model's weight

adjustments as well as strengths and weaknesses. These optimisers are like training coaches for the model, and some common ones include Stochastic Gradient Descent (SGD), Adam, RMSprop, etc. The choice of optimiser can impact how quickly and effectively CNN learns and converges to an optimal model. Optimiser is a key component in the training of a CNN, ensuring that the model's parameters are updated in a way that minimises the loss and helps the model improve its performance on the given task. The selection of the optimiser can have a significant impact on the training process and the ultimate quality of the model.

5.3 Importance of Hyperparameter Optimisation

Optimal hyperparameters are paramount for achieving the best model performance (Feurer & Hutter, 2019). Inappropriately chosen hyperparameters may lead to suboptimal model training, while using a limited number of optimal hyperparameters in machine learning can result in poor network performance. Striking a balance between model complexity and performance in alignment with available resources is essential.

A systematic exploration is crucial to optimise hyperparameters, focusing on achieving improved model performance while judiciously utilising time and computational resources. Various optimisation algorithms are available to determine the optimal conditions that enhance model performance.

5.4 Hyperparameter Optimisation Algorithms

The right set of hyperparameters that lead to effective performance can be achieved by employing an appropriate search strategy. We use grid and random search techniques to find the hyperparameters that optimise the CNN model.

Grid Search: It is a method of optimising a model that systematically selects hyperparameters. This approach explores all possible combinations within the search space without considering their effects on the optimisation process. Each parameter is treated with equal probability to influence the optimisation. While this method guarantees that no hyperparameter combination is overlooked (Ali et al., 2022), it demands more time and computational resources due to the exhaustive exploration of all combinations (Zhang et al., 2019).

Random Search: Unlike grid search, random search does not explore all possible combinations. Instead, it optimises hyperparameters by randomly selecting combinations. Since random search does not cover every point in the search space, it typically requires fewer computational resources and less time to identify optimal hyperparameters. Random search can be preferred when considering the trade-off between performance and resource constraints. Random search is particularly effective in large-scale problems compared to deterministic algorithms, and it proves to be invaluable in tackling complex problems with black-box function evaluation (Zabinsky, 2009).

5.5 Experiment Setup

In hyperparameter optimisation, we explore different values of every hyperparameter in the defined search space, measuring the model's performance for each value of the hyperparameter. The hyperparameter value giving the best performance is recorded. The usual data preparation steps for training the CNN models have been followed, such as data collection and pre-processing, including data selection, cleansing, undersampling, encoding, and splitting. The two algorithms – grid search and random search – are used to optimise the CNN model.

Data Collection:

Collecting real fraudulent transaction data is a challenging task. Firstly, this financial data is very sensitive, and organisations do not want to disclose it. Secondly, if such data is disclosed to others, the customers of the organisation would lose confidence in them. In this case, synthetic data could be useful for training machine learning models without compromising customers' privacy. The synthetic data provided by Erik Altman in https://www.kaggle.com/datasets/ealtman2019/credit-card-transactions mimics realistic data that includes customer information, card details, and transaction information. The following three data tables contain the dataset: '*sd254_users'*, '*sd254_cards'*, and '*credit_card_transactions-ibm_v2'*, as shown in Figure 5.2.

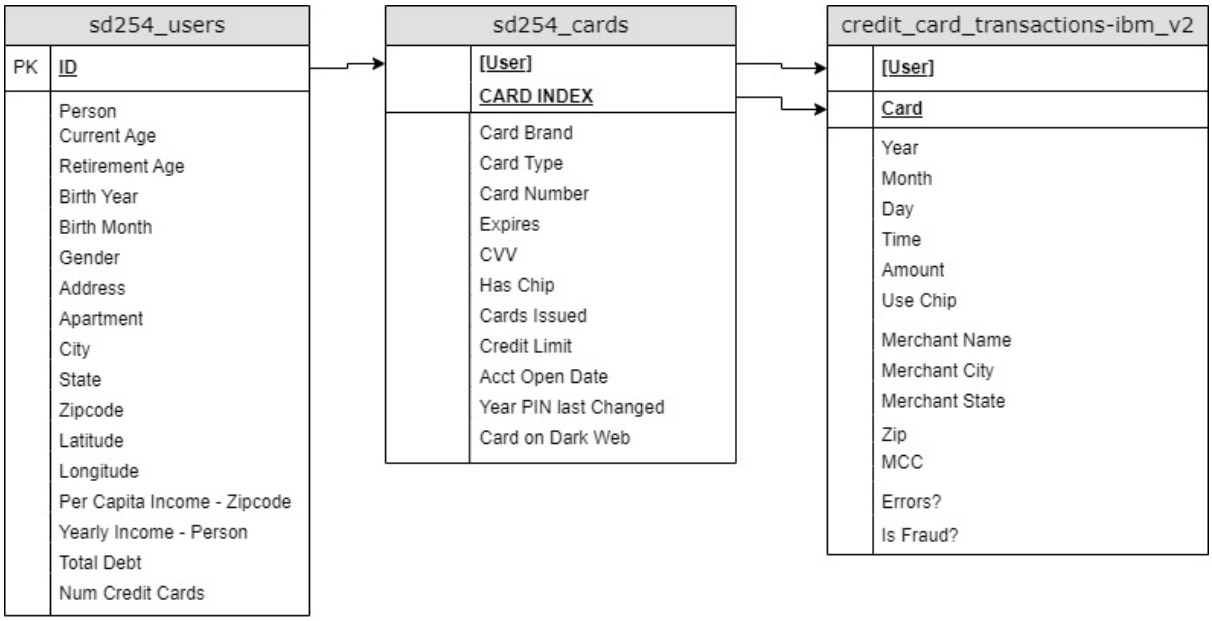


Figure 5-2 Entity-Relationship Diagram of Tables in the Fraudulent Transaction Dataset.

Table 5-1 illustrates the statistics. The transaction data table defines whether data is fraudulent or legitimate. For the convenience of using the data for model training, we have created a dataset by joining all three tables, producing a larger table containing all columns. We added a new column, 'ID,' in the 'sd254_users' table, which is used for joining the 'sd254_cards' table using the 'user' column as the common key. Additionally, the 'user' and 'card index' fields from the 'sd254_cards' table are linked to the 'user' and 'card' columns of the 'credit_card_transactions-ibm_v2' table, respectively.

Table 5-1 Fraudulent Transaction Dataset Information

Table name	**Related to**	**Number of records**	**Number of Features**	**Fraud records**	**% Fraud Transaction**
credit_card_transactions-ibm_v2	Customer information	24386900	15	29737	0.12%
sd254_cards	Card information	6146	13	-	-
sd254_users	Transaction information	2000	18	-	-
Combination of all tables	Combined information	10588051	45	12739	0.12%

Data preprocessing:

- *Data Selection:* We have extracted a total of 10,588,051 records from an initial pool of 24,386,900 records. This selected portion of data has the same ratio of fraudulent records (i.e., 0.12%) as in the *'credit_card_transaction_ibm_v2'* table and the combined table.
- *Data Cleansing:* null data are replaced with zero, and invalid characters, such as the dollar sign beside the transaction amount, are removed at this stage.
- *Data Undersampling:* The dataset is found to be highly imbalanced, where the ratio of legitimate transaction records (10588051) to fraudulent transaction records (12,739) in the dataset is 99.88:0.12. To balance the majority class with the minority one, an undersampling method is used. In the undersampling method, the number of records randomly selected from the majority class is equal to the number of records originally in the minority class, which equalises the majority and monitory class sample sizes with the same number of records, which is 12,739.
- *Data Encoding:* Categorical features or text type fields are transformed into numerical values using Label Encoder.
- *Data Splitting:* The dataset is split into training and test datasets with an 80:20 ratio; 20% of the data from the training dataset was used as a validation dataset.

Hyperparameter Optimisation:

We begin with a baseline CNN model incorporating the default settings for hyperparameter tuning. This baseline model is constructed, trained, and tested using the dataset outlined in the preceding section. The hyperparameters employed in the baseline CNN model are detailed in Table 5-2.

Table 5-2 Hyperparameter Values Used in the Baseline Model

Name of Hyperparameters	Value of Hyperparameters
Number of Convolutional Layers	1
No of Filters	4
Filter/Kennel size	3
Number of Pooling Layers	1
Pool Size	2
Learning Rate	0.0001
Activation function	Relu
Dropout rate	0
Optimiser	Adam

In the following step, we specify the number of hyperparameters we need to adjust to optimise the CNN model. Next, we define the search space for each hyperparameter, which is given in Table 5-3.

Table 5-3 Search Space for CNN Hyperparameter Optimisation

Step	Hyperparameters	Search Space
Step-1	Number of Convolutional Layers	[1, 3, 5, 7,9]
Step-2	Number of Filters	[4, 8, 16, 64]
Step-3	Filter/Kernel Size	[3, 5, 7]
Step-4	Learning Rate	[0.001, 0.0001, 0.00001]
Step-5	Optimiser	['adam', 'sgd', 'rmsprop', 'adagrad', 'adadelta']
Step-6	Number of Pooling Layers	[0, 1, 2, 3]
Step-7	Dropout Rate	[0.0, 0.2, 0.4,0.50, 0.6]
Step-8	Batch Normalisation	[True, False]
Step-9	Activation Functions	['relu', 'tanh', 'sigmoid']

We initiate the tuning process by selecting the first hyperparameter to be tuned, which in this case is the ‘Number of Convolutional Layers’. We apply the optimisation algorithm to explore this search space and update the model with the best-performing hyperparameter value. In the hyperparameter optimisation stage, training and testing occur to reach optimal performance.

In the first step, the optimum value for the ‘Number of Convolutional Layers’ is selected, which is 3. In the subsequent step, we tune the second hyperparameter, which is the ‘Number of Filters’, while keeping the value for the first hyperparameter constant at the previously found optimised value. We repeat this process for all successive

hyperparameters. Ultimately, after tuning all hyperparameters, we arrive at the final optimised CNN model.

We use both the random and grid search algorithms to explore the search space and find the optimum value for each hyperparameter. The whole process of hyperparameter optimisation is illustrated in the flowchart of Figure 5-3.

Model Testing and Evaluation:

The performance of the model optimised by hyperparameters that are found with grid search and random search is measured in terms of the following indexes: Precision, Recall, F1-score, and Area under the ROC curve (AUC), which have been broadly discussed in Chapter 3 and Chapter 4.

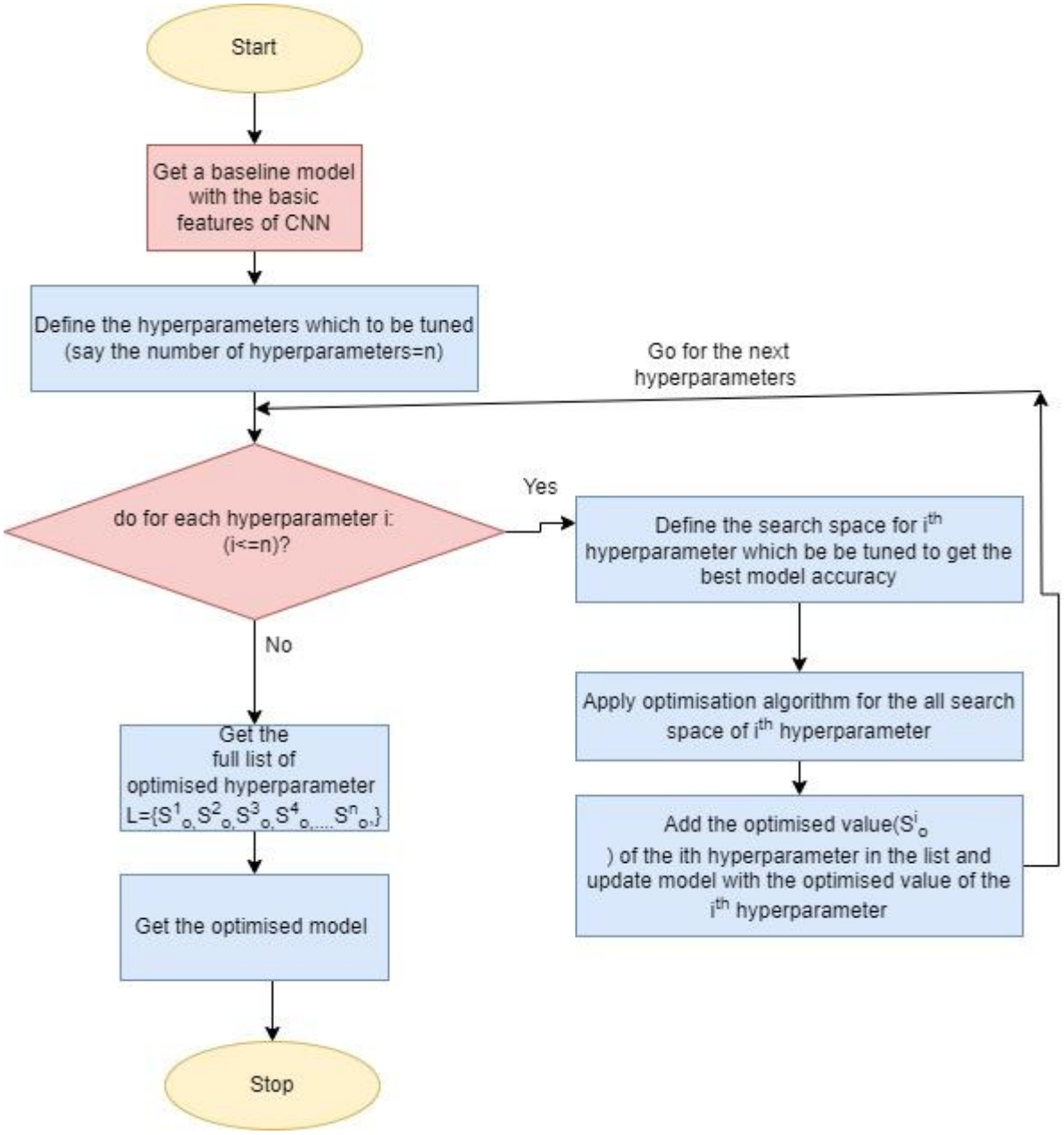


Figure 5-3 Flowchart for the Hyperparameter Optimisation of CNN.

5.6 Experimental Results

This section analyses the performance of the CNN models optimised with grid search and random search in detecting fraudulent transactions. The performance of the optimised CNN models is also compared against the baseline CNN model. Finally, the performance of the optimised CNN models is compared against a state-of-the-art method proposed by Breskuvienė and Dzemyda (2023). To the best of our knowledge, this is the

only state-of-the-art method using the fraudulent transaction dataset considered in this chapter.

Table 5-4 displays the hyperparameter values of the CNN model pre and post-optimisation. The baseline CNN model has only 1 convolutional layer. The random search optimised CNN model consists of 3 convolutional layers, while the grid search optimised one has 5 layers. Increasing the number of layers allows the model to learn more complex features but can lead to overfitting issues. In the baseline CNN model, only four filters exist in each convolutional layer. However, optimisation using random and grid search algorithms increased the number of filters to 64 and 16, respectively. This increase in the number of filters enables the CNN model to capture diverse features from the data effectively. The baseline CNN model includes one pooling layer, but both optimised CNN models exclude the pooling layers altogether. This omission suggests that the optimised CNN models rely on the original feature vectors extracted by the convolutional layers without further downsizing them. The learning rate differs significantly among the models: 0.0001 for the baseline CNN model, 0.001 for the CNN model optimised by random search, and 0.00001 for the CNN model optimised by grid search. Smaller learning rates are more efficient for model training and convergence but may require more time to reach the optimal point. The activation function, optimiser, and batch normalisation settings remain consistent across all models, with 'relu,' 'adam,' and 'false' being used, respectively. Dropout is introduced into the optimised CNN model by random and grid searches at 0.2 and 0.5, respectively. This addition helps reduce the CNN model's susceptibility to overfitting.

Table 5-4 Hyperparameters of the Base Line and Optimised CNN Model

Name of Hyperparameters	Base Line Model	Optimised CNN by Random Search	Optimised Grid Search
Number of Convolutional Layers	1	3	5
No of Filters	4	64	16
Filter/Kennel size	3	3	3
Number of Pooling Layers	1	0	0
Pool size	2	0	0
Learning Rate	0.0001	0.001	0.00001
Activation Function	relu	Relu	relu
Dropout Rate	0	0.2	0.5
Optimiser	Adam	Adam	Adam
Batch Normalisation	FALSE	FALSE	FALSE

Optimised Model Validation:

Accuracy and loss curves are essential tools for monitoring and evaluating the performance of machine learning models, particularly during training. They provide insights into how well a model learns from the training data and how well it generalises to unseen data. These curves are typically plotted as a function of epochs (training iterations) across the training phase. In the optimisation process, we used nine steps. Four steps of tuned hyperparameters were found as a baseline model, which are Step-5 (Optimiser), Step-6 (Number of Pooling Layers), Step-8 (Batch Normalisation) and Step-9 (Activation Functions). Thus the tuning model for optimisation of the rest of the five steps has been explained in the next part of this section.

We allocate 20% of the training data for validation purposes at each step of optimisation. We also employ an early stopping callback with a patience of 5, which means the training process ends if the performance does not improve for five consecutive epochs.

Training and Validation of Model Optimised by Random Search:

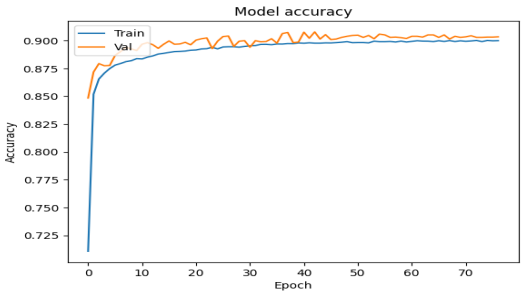
The training and validation accuracy as well as loss curves for the baseline and the optimised CNN models by random search technique, are depicted in Figure 5-4. As shown in Figure 5.4(a1), the training accuracy of the baseline CNN model steadily increases until epoch number 40, but remains relatively constant for the remainder of the epochs. However, the validation accuracy curve exhibits rapid fluctuations in every epoch, with the validation accuracy curve being consistently above the training accuracy curve. This suggests that the baseline CNN model performs well on the training data but struggles to generalise to the validation data. Figure 5.4(a2) shows that the training loss for the baseline CNN model steadily decreases until epoch number 40 but remains relatively constant afterwards. Although the validation loss curve fluctuates frequently, it remains consistently below the training loss curve.

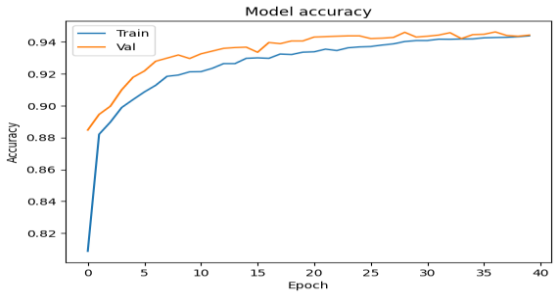
We have followed the steps as described in Table 5-3. In the first step, we use feature space [1, 3, 5, 7, 9] to tune the model in the number of layers by deploying a random search algorithm. Model performance is optimal when number of convolutional layers is 3. In this step, the validation curve remains above the training curve, but the

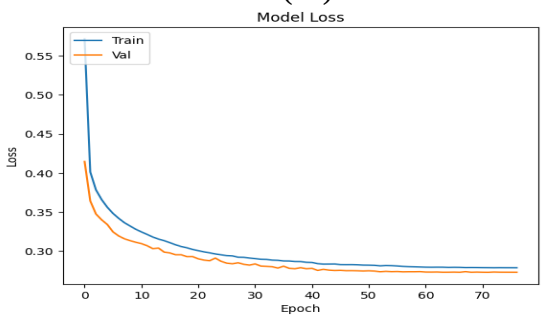
performance improves for both the validation and training datasets. Additionally, the loss curve shows a decreasing trend for the training and validation dataset, as seen in Figures 5.4 b(1) and 5.4 b(2).

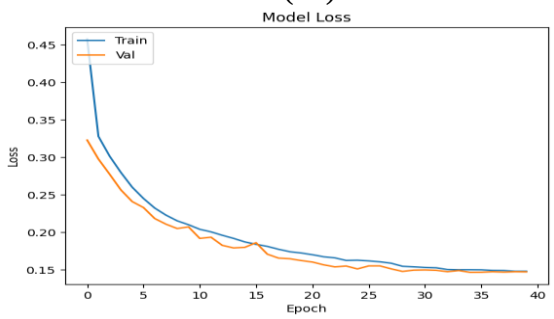
In the second step, we tune the model using the feature space of [4, 8, 16, 64] filters. The model shows the best performance with 64 filters. In this step, performance improves slightly. However, the between the validation and training curves increases with the progress of epochs, indicating a potential overfitting problem, as shown in Figures 5.4 (c1) and 5.4 (c2).

By optimising the learning rate, it is observed that the model reaches convergence in just 16 epochs, which is faster than the previously optimised model, as shown in Figures 5.5 (d1) and 5.5 (d2). Finally, introducing dropout layers reduces the gap between the validation and training curves in both accuracy and loss graphs in Figures 5.5 (e1) and 5.5 (e2).

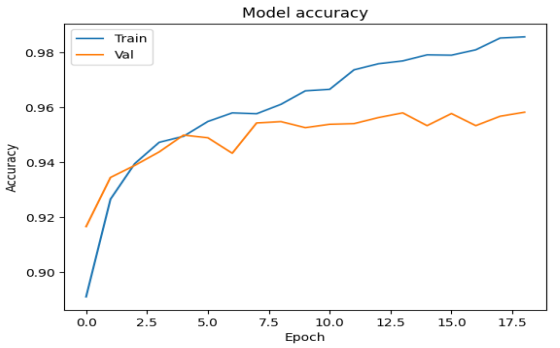
<table>
<tr><td>For Baseline Model</td><td>For Convolution Layer: 3</td></tr>
<tr><td> 

5.4 (a1)</td><td> 

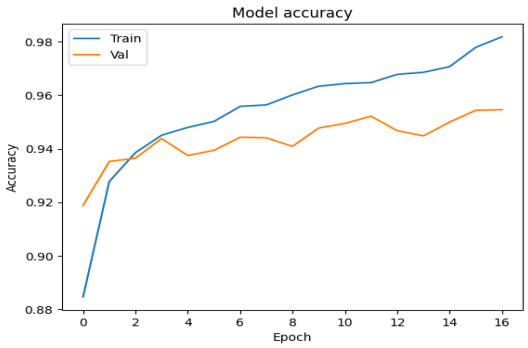
5.4 (b1)</td></tr>
<tr><td> 

5.4 (a2)</td><td> 

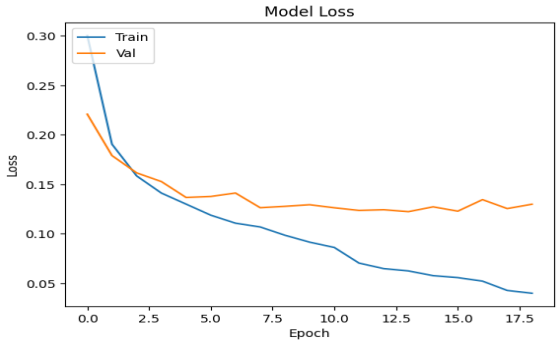
5.4 (b2)</td></tr>
<tr><td>For Filter: 64, filter Size: 3</td><td>For Learning Rate: 0.001</td></tr>
</table>



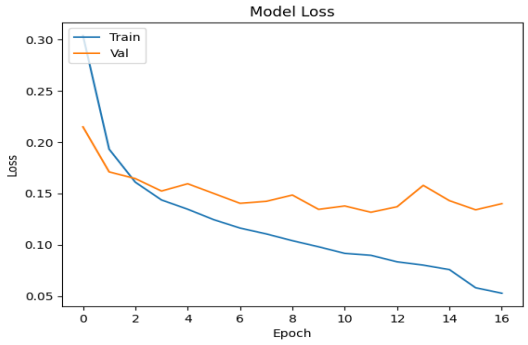
5.4(c1)



5.4 (d1)

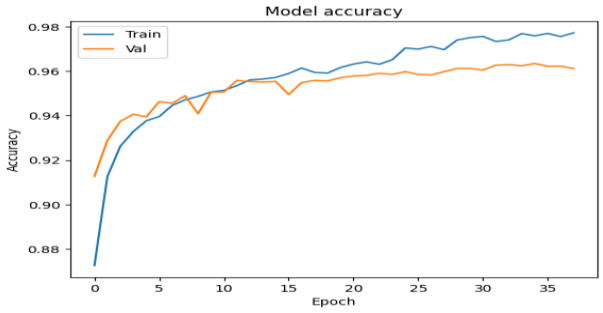


5.4 (c2)

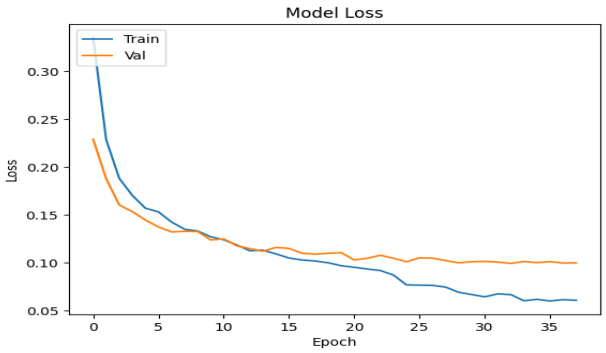


5.4 (d2)

For Dropout Rate: 0.2



5.4 (e1)



5.4 (e2)

Figure 5-4 The Figure Displays 5 Pairs of Graphs: Accuracy vs. Epoch and Loss vs. Epoch. Each pair represents a different model stage using a Random Search Algorithm: Baseline (5.4(a1, a2)), Convolutional Layers 5.4(b1, b2), Filters 5.4(c1, c2), Learning Rate 5.4(d1, d2), and Dropout Rate 5.4(e1, e2).

Training and Validation of Model Optimised by Grid Search:

The accuracy and loss curves against every epoch are given in Figure 5.5. Starting with 5.5(a1, a2), which is shown for the baseline model, the curve for validation data fluctuates over the epochs and is above the training curve (see Figure 5.5(a1)). The loss curve is very similar for both the training and validation datasets.

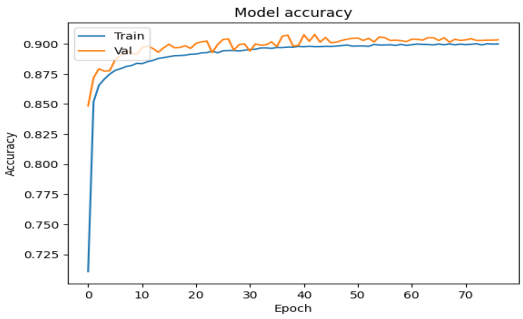
Then, in the tuning process, the model performed best with a Convolution Layer of 5. The accuracy vs. epoch curve (5.5(b1)) shows that accuracy increases from the baseline model. The accuracy curve for training data is still below the validation curve, indicating that the model is facing an under-fitting problem. The loss curve quickly reaches the saturation point in epoch 38, indicating that the model's learning capacity improved at this step (see Figure 5.5(b2)).

After this step, in Figure 5.5(c1, c2), the model performs better with a size of 16 and filter size 3. The accuracy improves, and the model gets trained more quickly than in the previous step, reaching epoch 25. However, the gap between the validation and training curves increases.

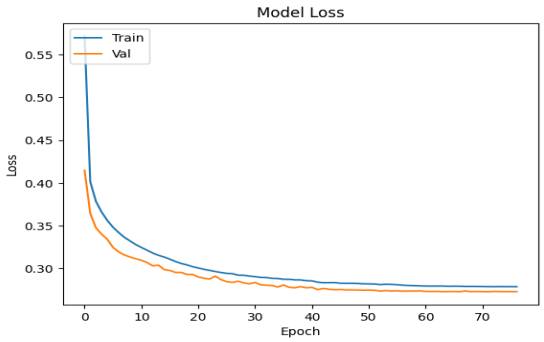
Moving on to the next step in the optimisation process with the learning rate, we see that a learning rate of 0.00001 fits the model best. The accuracy and loss curves in Figure 5.5(d1, d2) show that, although accuracy significantly improves, the gap between the training and validation curves reduces slightly.

In the last stage of the optimisation process, the gap between the validation and training curves almost disappears by dropping 50% of nodes after the previous convolutional layer. This indicates that the model becomes more generalised and overcomes the overfitting problem when hyperparameters are optimised. It is expected to perform well on unseen data, which we demonstrate in the testing phase of the model evaluation (see Figure 5.5(e1, e2)).

For Baseline Model

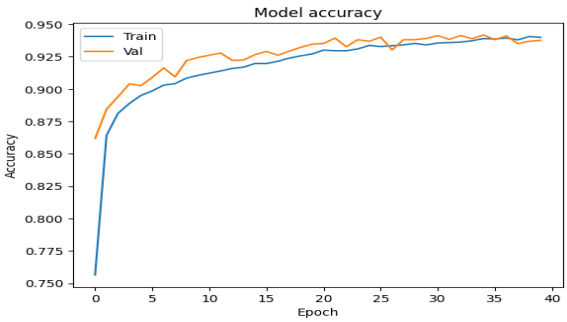


5.5 (a1)

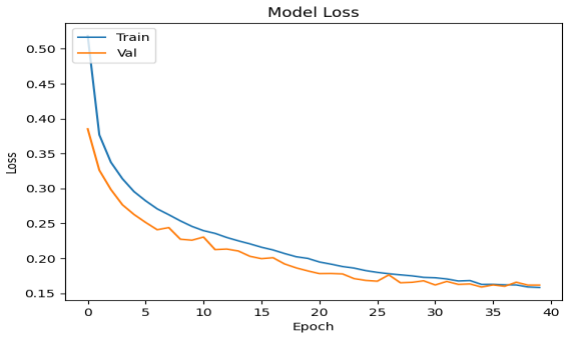


5.5 (a2)

For Convolution Layer: 5

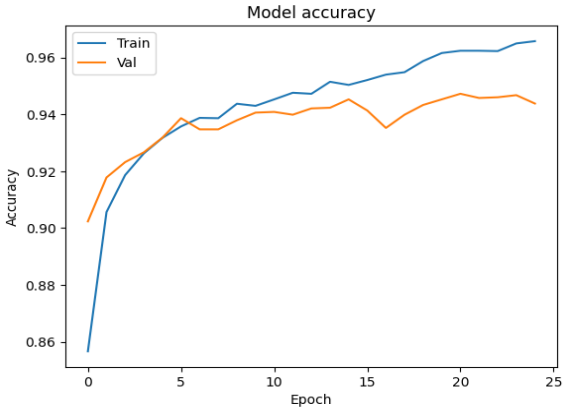


5.5 (b1)

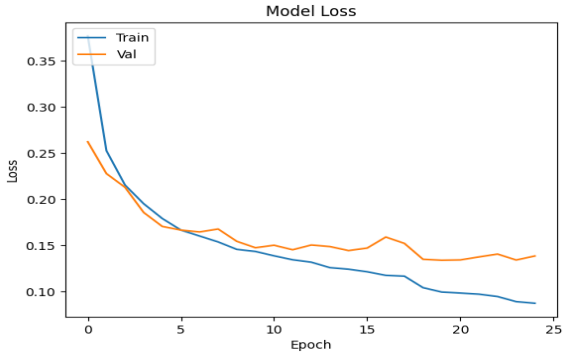


5.5 (b2)

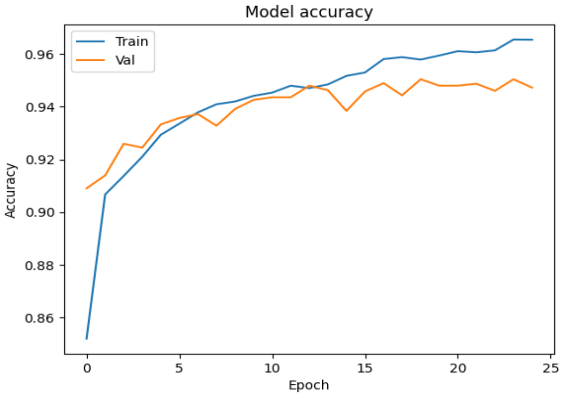
For Filter: 16, filter Size: 3



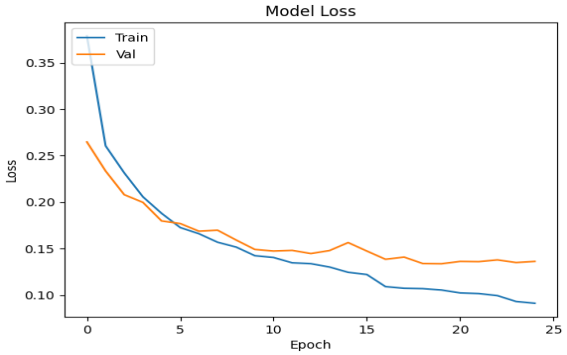
5.5(c1)



For Learning Rate: 0.00001

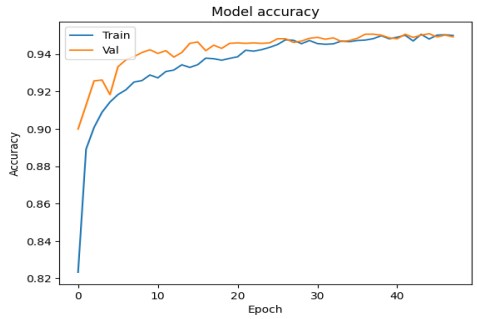


5.5 (d1)

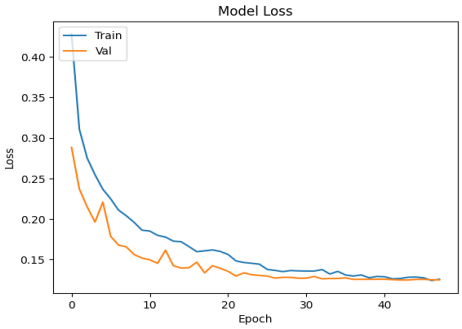


5.5 (c2)	5.5 d(2)

For Dropout Rate: 0.5



5.5 (e1)



5.5 (e2)

Figure 5-5 The Figure Displays 5 Pairs of Graphs: Accuracy vs. Epoch and Loss vs. Epoch. Each pair represents a different model stage while using a grid search algorithm for Optimisation: Baseline (5.5(a1, a2)), Convolutional Layers 5.5(b1, b2), Filters 5.5(c1, c2), Learning Rate 5.5(d1, d2), and Dropout rate 5.5(e1, e2).

Model Evaluation with Test Data:

In our testing phase, we utilise 20% of the original dataset as a test dataset, entirely unseen by the CNN model during training. At each optimisation step, we rigorously evaluate the model's performance. We generate confusion matrices and AUC curves to visualise the progress of the optimisation process for both random search and grid search algorithms.

Evaluation of optimised model while random search algorithm is used:

In Figure 5.6(a1), the confusion matrix for the base CNN model illustrates that, out of 2,548 fraud transactions, 2,292 are correctly detected as fraud (true positives), while 256 are misclassified as non-fraud or legitimate transactions (false negatives). Further, 2,274 non-fraud transactions are correctly classified as non-fraud (true negatives), while 274 non-fraud transactions are misclassified as fraud transactions (false positives).

Figure 5.6(a2) shows the Receiver operating characters curve, where the AUC value for the baseline model is 0.9445.

In step 2 of the optimisation process, with the convolution layer 3 models from the confusion matrix shown in Figure 5.6(b1), false misclassification is reduced by 37% from 530 (=274 + 256) to 329 (=137+192), and the AUC value turns to 0.9831 (see Figure 5.6(a2)).

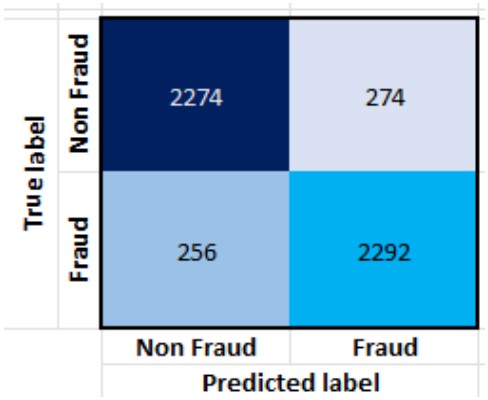
For filter number 64 and filter size 3, misclassification reduces to 21% from 329 (=137+192) to 260 (=161+99) (figure 5.6(c1)) and the AUC value increased to 0.9902 (Figure 5.6(c2)).

While tuning the model using learning rate = .001, the confusion matrix in Figure 5.6 shows that the misclassification rate reduces by 3% from 260(161+99) to 252 (=94+158) (Figure 5.6 d(1)). However, down trends are observed in this case, where the AUC also reduces slightly to 0.9898 (Figure 5.6(d2)).

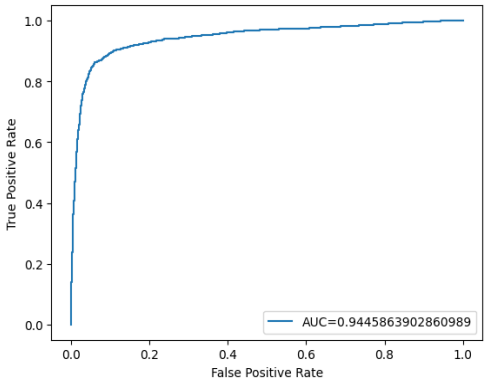
At the final step of the final stage of the optimisation process, comparing the confusion matrices of the baseline model in Figure 5.5(a1) and the final optimised model in Figure 5.5(e1), a substantial reduction in the number of fraud transaction misclassifications is observed: a drop from 256 to 118, representing a 53.6% improvement. Additionally, the number of non-fraud misclassifications falls from 274 to 112: a 59% reduction.

Moreover, the AUC for the optimised CNN model improves by 11% compared to the baseline CNN model, increasing from 0.8960 (see Figure 5.5(a2)) to 0.9913 (see Figure 5.5(e2)).

For Baseline Model

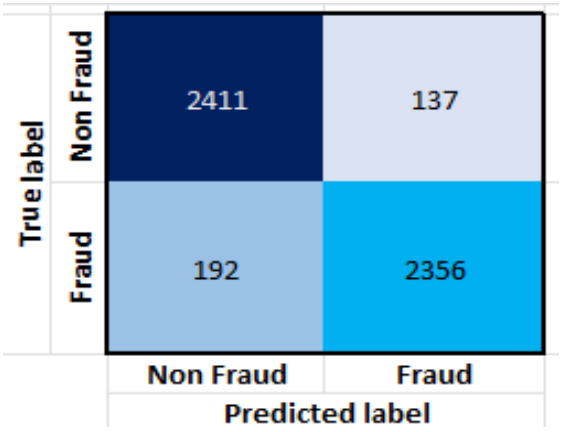


5.6 (a1)

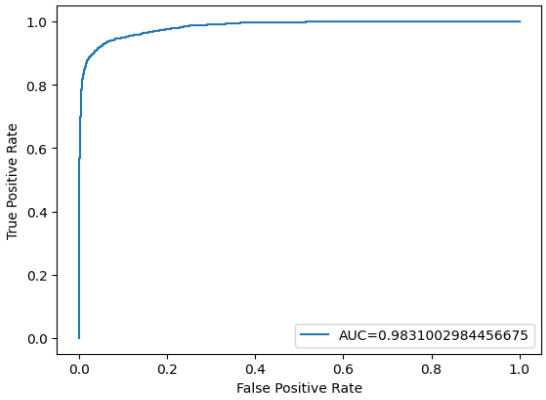


5.6 (a2)

For Convolution Layer: 3

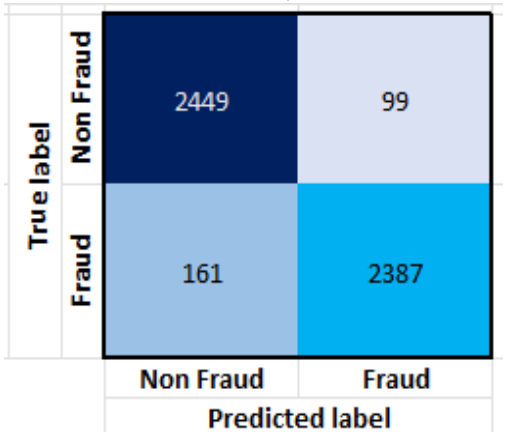


5.6 (b1)



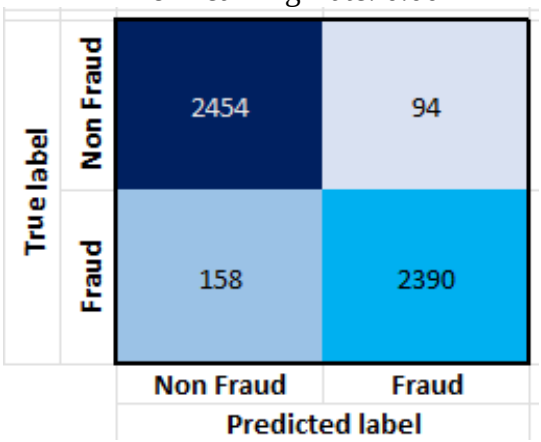
5.6 (b2)

For Filter: 64, Filter Size: 3

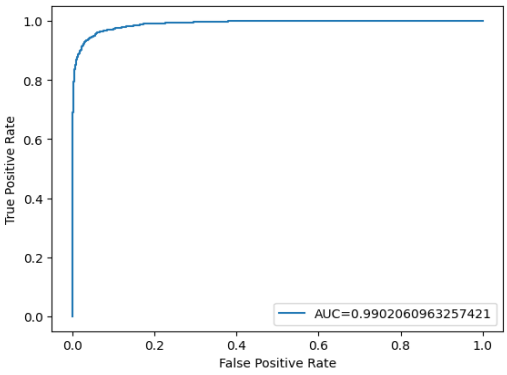


5.6 (c1)

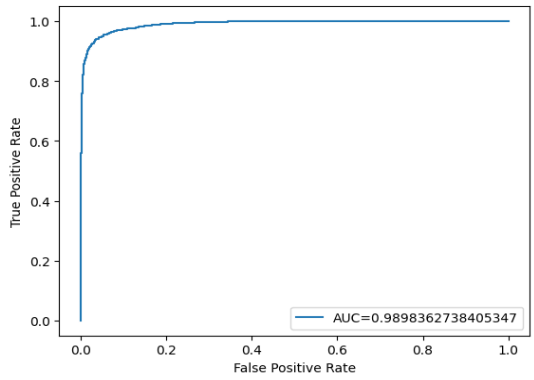
For Learning Rate: 0.001



5.6 d(1)

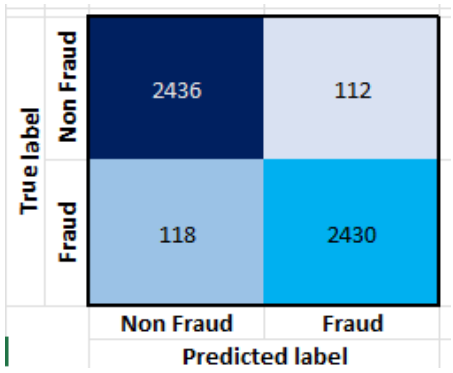


5.6 (c2)

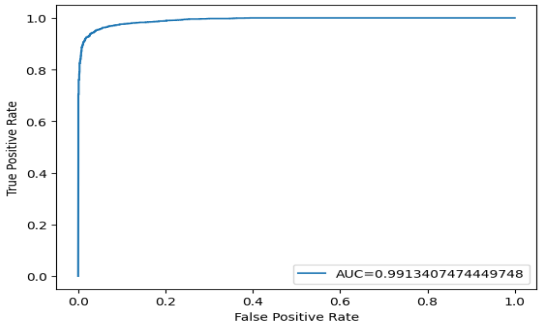


5.6 d(2)

For Dropout Rate: 0.2



5.6 (e1)



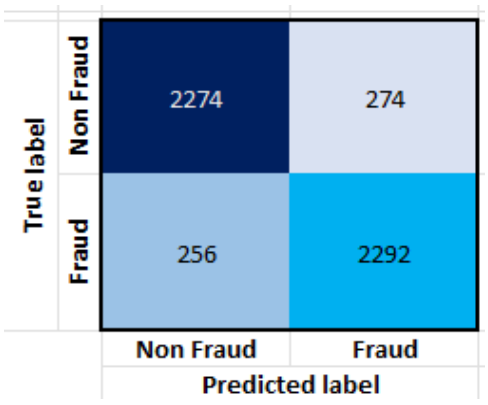
5.6 (e2)

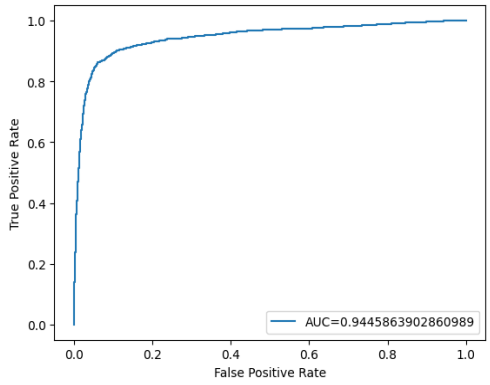
Figure 5-6 The Figure Displays 5 Pairs of Graphs: Confusion Matrix and AUC when a Random Search Algorithm is used. Each pair represents a different model stage: Baseline 5.6(a1, a2), Convolutional Layers 5.6(b1, b2), Filters 5.6(c1, c2), Learning Rate 5.6(d1, d2), and Dropout Rate 5.6(e1, e2).

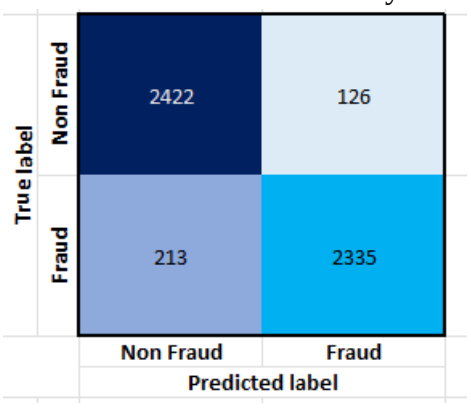
Evaluation of Optimised Model When Grid Search Algorithm is used:

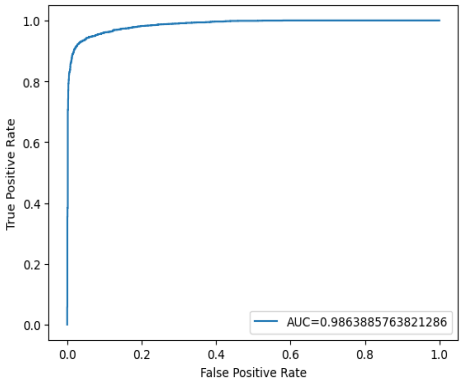
We follow a similar process for evaluating the optimised model, which is optimised by using grid search like random search. Firstly, the baseline model of this process is the same as that of the random search algorithm (Figure 5.7(a1, a2)), which rightly classifies 89% of samples (4566 (=2274+2292) of 5078 (=2274+274+256+2292)) (Figure 5.7(a1)), and AUC value is 0.9446 (Figure 5.7(a2)). Next, in the second step of

optimisation with convolutional layer 5, we see that the model classifies 93.35% of data samples (Figure 5.7(b1)), and the AUC value improved to 0.9863 (Figure 5.7(b2)). The accuracy rates for Figures 5.7(c1), 5.8(d1) and 5.7(e1) are 94.27%, 94.52% and 94.50%, respectively. The AUC values of these three steps are 0.9861, 0.9854 and 0.9876 (Figures 5.7(c2), 5.8(d2) and 5.7(e2)), respectively.

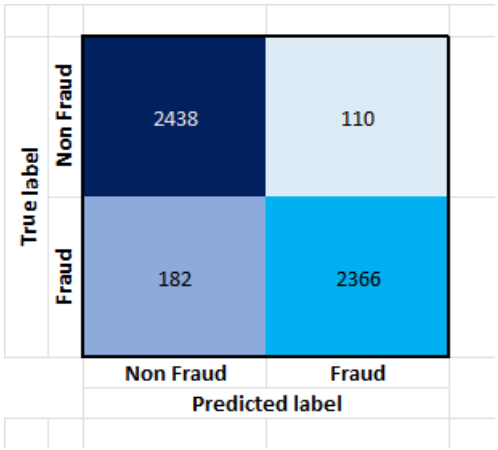
<table>
<tr>
<td>For Baseline Model
 

5.7 (a1)
 

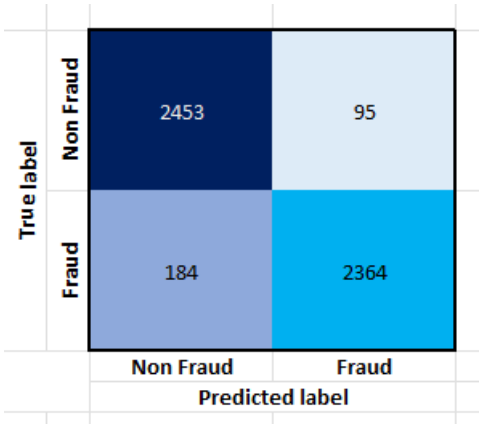
5.7 (a2)</td>
<td>For Convolution Layer: 5
 

5.7 (b1)
 

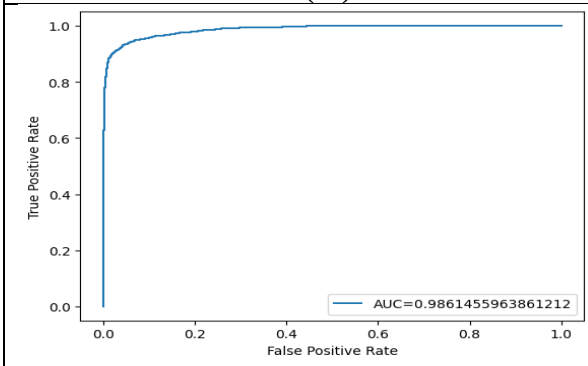
5.7 (b2)</td>
</tr>
<tr>
<td>For Filter: 64, Filter Size: 3</td>
<td>For Learning Rate: 0.00001</td>
</tr>
</table>



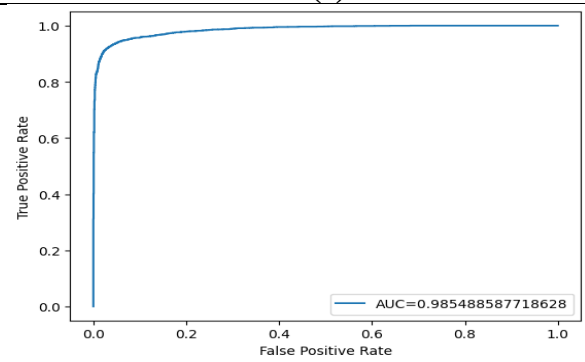
5.7 (c1)



5.7 d(1)

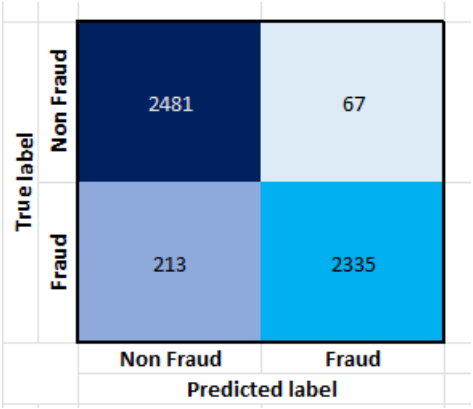


5.7 (c2)

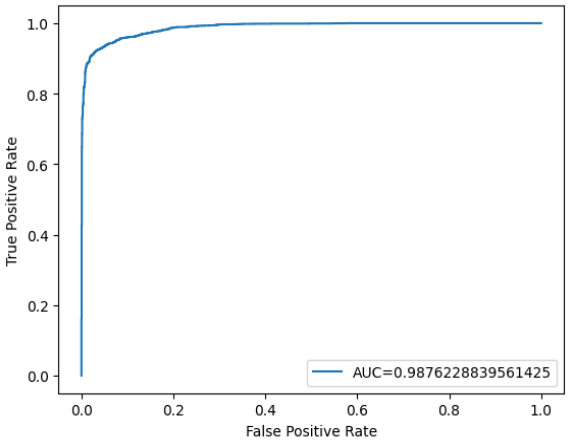


5.7 d(2)

For Dropout Rate: 0.5



5.7 (e1)



5.7 (e2)

Figure 5-7 The Figure Displays 5 Pairs of Graphs: Confusion Matrix and AUC while using Grid Search. Each pair represents different model stage: Baseline 5.7(a1, a2), Convolutional Layers 5.7(b1, b2), Filters 5.7(c1, c2), Learning Rate 5.7(d1, d2), and Dropout Rate 5.7(e1, e2).

The performance improvement in terms of precision, recall, F1-score, and AUC for the tuning of each hyperparameter is presented in Table 5.5. The corresponding data is also visualised in the bar chart in Figure 5.6. Notably, the performance significantly improves in the first step but exhibits a more consistent and gradual increase from the second to the ninth step. The overall performance in fraudulent transaction detection at the final optimisation level improves by 7%.

Table 5-5 Performance Evaluation of CNN Model with Hyperparameter Optimisation using Random Search Algorithm

SL	Optimisation Steps	Description	Accuracy	Precision	Recall	F1-Score	AUC
1	Base Line	Base Line Model	0.8960	0.8960	0.8960	0.8960	0.8960
2	Optimisation-1	Varying Convolutional Layers	0.9354	0.9356	0.9354	0.9354	0.9831
3	Optimisation-2	Varying Filter/Kernel Size and Numbers	0.9490	0.9492	0.9490	0.9490	0.9902
4	Optimisation-3	Varying Learning Rate	0.9505	0.9508	0.9505	0.9505	0.9898
5	Optimisation-4	Varying Optimiser	0.9508	0.9508	0.9505	0.9505	0.9898
6	Optimisation-5	Varying Number of Pooling Layers	0.9508	0.9508	0.9505	0.9505	0.9898
7	Optimisation-6	Varying Dropout Rate	0.9549	0.9549	0.9549	0.9549	0.9913
8	Optimisation-7	Batch Normalisation used	0.9549	0.9549	0.9549	0.9549	0.9913
9	Optimisation-8	Varying Activation Function	0.9549	0.9549	0.9549	0.9549	0.9913
		% of Increase in the Optimised model	7%	7%	7%	7%	11%

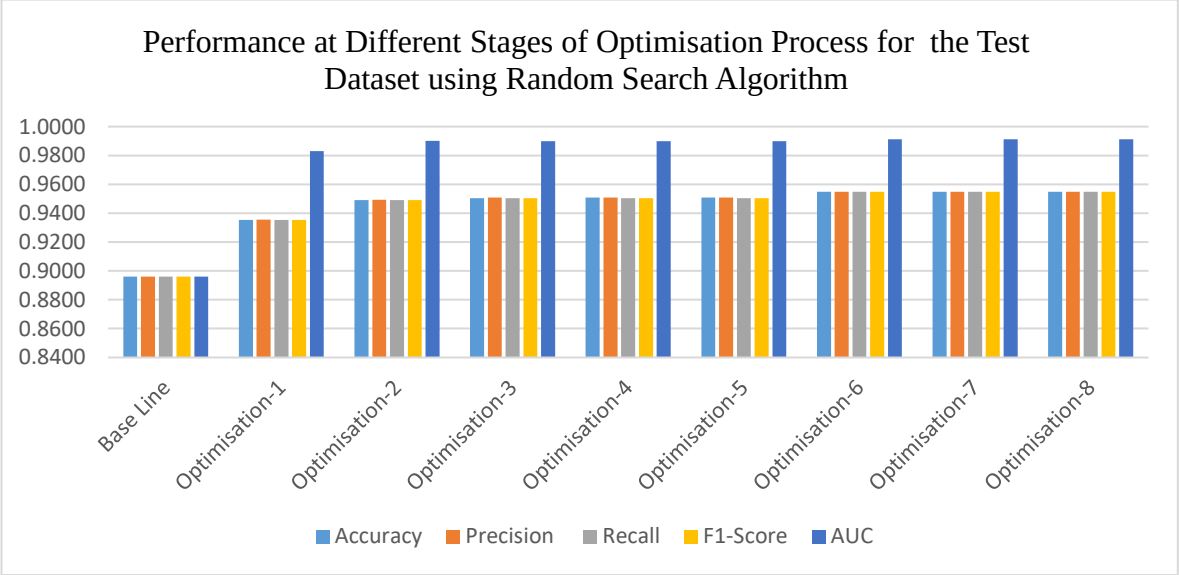


Figure 5-8 Performance Evaluation of CNN Models with Hyperparameter Optimisation using a Random Search Algorithm.

Similarly, the performance improvement in terms of precision, recall, F1-score, and AUC for the tuning of each hyperparameter for using grid search is presented in Table 5.6. The corresponding data is also visualised in the bar chart in Figure 5.9. Notably, the performance significantly improves in the first step but exhibits a more consistent and gradual increase from the second to the ninth step. The accuracy, precision,

recall, F1 and AUC performance measurement indexes in fraudulent transaction detection at the final optimisation level improve by 5.47%, 5.64%, 5.48%, 5.47% and 10.13%, respectively.

Table 5-6 Performance Evaluation of CNN Model with Hyperparameter Optimisation Using Grid Search Algorithm

SL	Optimisation Steps	Description	Accuracy	Precision	Recall	F1-Score	AUC
1	Base Line	Base Line Model	0.8960	0.8960	0.8960	0.8960	0.8960
2	Optimisation-1	Varying Convolutional Layers	0.9335	0.9340	0.9335	0.9335	0.9863
3	Optimisation-2	Varying Filter/Kernel Size and Numbers	0.9427	0.9431	0.9427	0.9427	0.9861
4	Optimisation-3	Varying Learning Rate	0.9453	0.9458	0.9453	0.9452	0.9855
5	Optimisation-4	Varying Optimiser	0.9453	0.9458	0.9453	0.9452	0.9855
6	Optimisation-5	Varying Number of Pooling Layers	0.9453	0.9458	0.9453	0.9452	0.9855
7	Optimisation-6	Varying Dropout Rate	0.9450	0.9465	0.9451	0.9450	0.9868
8	Optimisation-7	Batch Normalisation used	0.9450	0.9465	0.9451	0.9450	0.9868
9	Optimisation-8	Varying Activation Function	0.9465	0.9465	0.9451	0.9450	0.9868
		% Increased in the Optimised model	5.47%	5.64%	5.48%	5.47%	10.13%

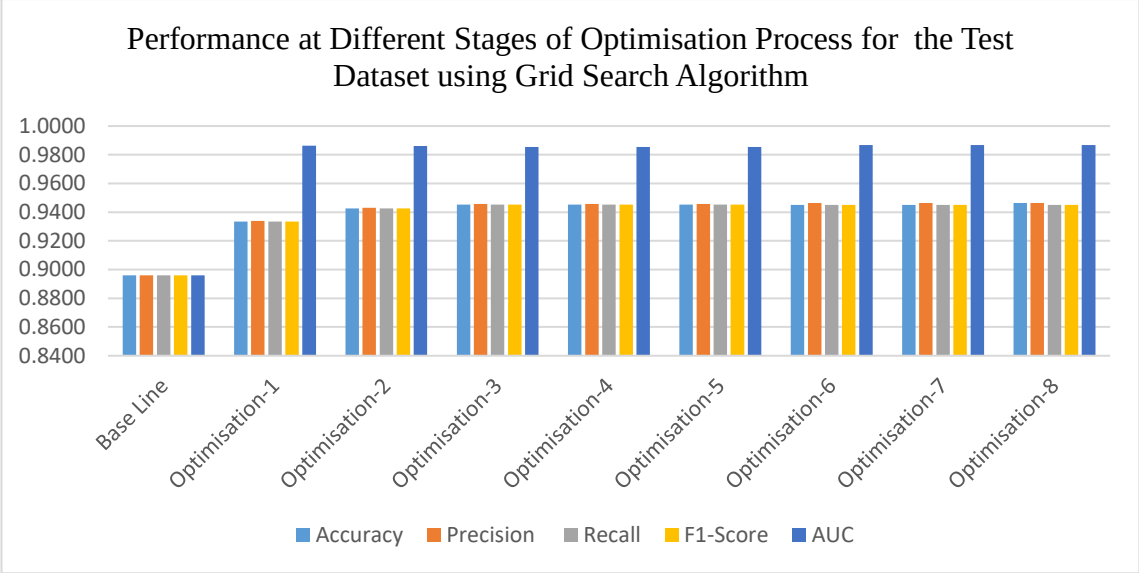


Figure 5-9 Performance Evaluation of CNN Model with Hyperparameter Optimisation using a Grid Search Algorithm

Table 5-7 displays the performance indices for both grid search and random search, while Figure 5-10 shows the corresponding bar graph. Notably, random search outperforms grid search across all performance indexes of fraudulent transaction detection, albeit by a small margin.

Table 5-7 Performance Comparison between the CNN Model Optimised with Random Search and Grid Search Algorithms

Algorithm	Accuracy	Precision	Recall	F1-Score	AUC
Random Search	0.9549	0.9549	0.9549	0.9549	0.9913
Grid Search	0.9465	0.9465	0.9451	0.9450	0.9868

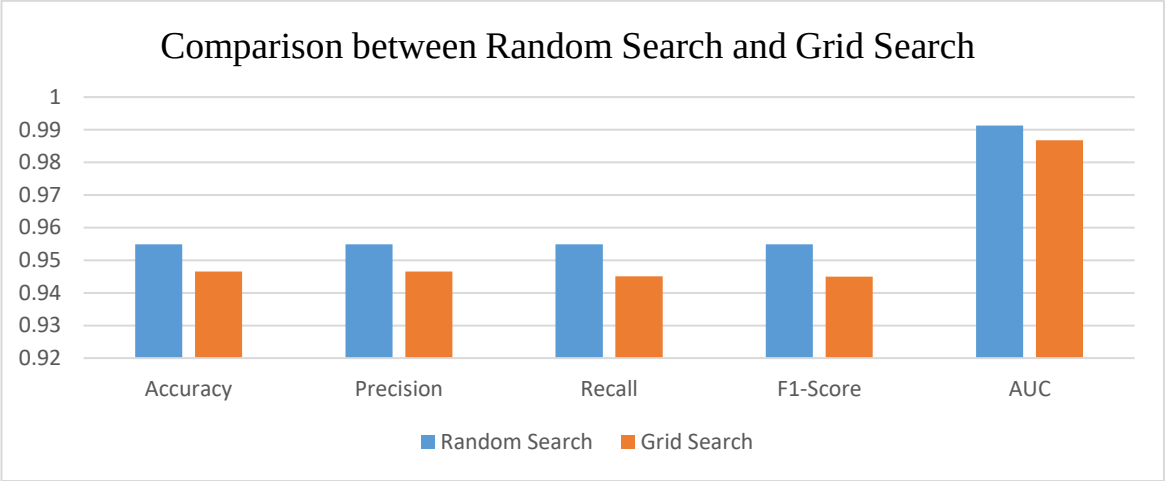


Figure 5-10 Performance Comparison between the CNN Model Optimised with Grid Search and Random Search Algorithms.

The performance of the optimised CNN models is compared against the state-of-the-art method proposed by Breskuvienė and Dzemyda (2023), who used the same dataset for performance evaluation. The comparison is presented in Table 5-8.

Table 5-8 Performance Comparison of the Optimised CNN Model against that of a State-of-the-Art Method

Performance Criteria	**Baseline CNN Model**	**State-of-the-art method**	**CNN Model Optimised with Random Search**	**CNN Model Optimised with Grid Search**	**Improvement for the CNN model optimised with random search compared to state-of-the-art method**	**Improvement for the CNN model optimised with grid search compared to state-of-the-art method**
Column→	**A**	**B**	**C**	**D**	**(C-B)*100/B**	**(D-B)*100/B**
Recall	0.8960	0.8670	0.9549	0.9443	10%	8.9%
False positive rate	10%	13%	4%	8%	9%	5%

Breskuvienė and Dzemyda (2023) aimed to enhance the recall rate and reduce false positive alarms. They employed a cluster-based classification method in their approach, and for unbalanced mitigation, they utilised undersampling. Their proposed method improved the recall rate from 0.845 to 0.867 and reduced the false positive rate

by 13.9%, from 323 to 278, compared to a baseline method mentioned in their paper. Our baseline and optimised CNN models are trained and evaluated on the same dataset as the one used by Breskuvienė and Dzemyda (2023). The CNN model optimised using random search improves the recall rate from 0.896 to 0.9549 and reduces the false positive rate by 9%, while the model optimised using the grid search improves the recall rate from 0.896 to 0.9443 and reduces the false positive rate by 5% compared to state-of-art-method.

5.7 Conclusion

In this chapter, we have explored various hyperparameters that significantly impact the performance of a CNN model to improve performance in fraudulent transaction detection. Additionally, we have discussed the crucial steps of model optimisation, which include building a baseline CNN model, defining the search space, selecting the optimisation algorithm, and outlining the step-by-step optimisation process. We have used two algorithms – grid search and random search – to find optimal hyperparameter values.

After performing optimisation, we evaluated the performance of the models in terms of accuracy, precision, recall, F1 score, and AUC to determine their effectiveness in fraudulent transaction detection. The optimised CNN model has demonstrated a 7% improvement over the baseline CNN model in terms of precision, recall, and F1 scores for the random search, while this improvement is about 5.5% for the grid search. Additionally, the improvement in AUC is 11% for the optimised CNN model compared to the baseline CNN model. The optimised CNN models have also outperformed a state-of-the-art fraudulent transaction detection method by a margin of 10% in terms of recall rate for the model optimised using random search and by a margin of 8.9% in terms of recall rate for the model optimised using grid search. Therefore, hypothesis H2 is validated.

The efficacy of using the optimised hyperparameters suggests that the insertion of more convolutional layers, more filters, and the inclusion of neuron dropout contributes to improving CNN models. ReLU turns out to be the most suitable activation function,

while Adam turns out to be the best optimiser. Although pooling layers could potentially reduce the training time, the random and good search algorithms omitted pooling layers to achieve higher precision and recall.

In the following chapter, we summarise the overall project findings, state the limitations of our study and discuss future research directions for the study.

Chapter 6. Conclusions

6.1 Introduction

This chapter offers a comprehensive overview of the entire thesis. It succinctly restates the research questions, discusses how these questions are addressed, and summarises the methodology employed in the research. Furthermore, it provides a summary of the application of feature engineering and optimisation algorithms to enhance the performance of the deep learning model. Finally, the chapter addresses the study's limitations and outlines potential future directions for research.

6.2 Research Summary

The thesis presents a study on detecting fraudulent transactions, utilising deep learning techniques, with a particular focus on the Convolutional Neural Network (CNN) models. Initially, a feature engineering technique, Deep Feature Synthesis (DFS), was employed to enhance the performance of CNN. Subsequently, CNN's performance was further optimised through hyperparameter tuning. The research contributions made in this thesis are outlined in the subsequent sections. The research questions addressed in this thesis are restated as follows.

1. How is deep learning (DL) used to identify and prevent fraudulent transactions?
2. What features of financial transactions are used to train the DL models?
3. How many hidden layers, and what filters give optimal results?
4. What is the best DL model for classifying fraudulent transactions?

6.3 Addressing Research Questions from Literature Review

We have conducted a thorough systematic literature review to find answers to the research questions. The detailed methodology of the literature review and the findings have been discussed in Chapter 2. The answers to the research questions are provided in Table 6-1.

Table 6-1 Research Questions Reconciliation

Research Questions	Research Questions' Answer	Details Found in
RQ1. How is deep learning used to identify and prevent fraudulent transactions?	Deep learning techniques for fraudulent transaction detection generally use the following six steps: 1. Data collection, 2. Data pre-processing, 3. Data pre-analysis, 4. Data splitting into training and testing sets, 5. Model training & optimisation, and 6. Model testing/evaluation. Since electronic transaction datasets are generally imbalanced, oversampling, undersampling or other techniques are used to mitigate the imbalance.	Section 2.4.1.1
RQ2. What features of financial transactions are used to train the DL models?	The features implicitly extracted from raw data by deep learning models such as CNN and autoencoders are used to train the classifier used for fraudulent transaction detection. While CNN can generate features at different levels of abstraction, the autoencoder can compress the representation of features by reducing the dimension.	Section 2.4.1.2
RQ3. How many hidden layers, and filters give optimal results for DL models?	The performance of CNN improves with the increase in the number of layers until a certain level, after which the model seems to overfit. The exact number of layers depends on the nature of the data. However, there is evidence of using 20 layers to achieve the optimum performance. Similarly, the more the number of filters, the more features are available to train a classifier, resulting in better performance.	Section 2.4.1.3
RQ4. What is the best DL for classifying fraudulent transactions?	The performance of the DL models varies across datasets, so it cannot be concluded that a particular model performs best in all situations. The CNN and autoencoder models outperform others for point data, while the RNN and LSTM models are used for sequence data. Using autoencoder and CNN in a pipeline, where CNN uses the features extracted by autoencoder, gives better results than passing raw data to CNN.	Section 2.4.1.4

6.4 Summary of Methodology

The research methodology behind the experiment design for validating the hypotheses discussed in Section 1.3.2 is detailed in Chapter 3 of the thesis. Those hypotheses have been refined in this chapter and are restated as follows:

H1: DFS improves the fraudulent transaction detection performance of CNN significantly.
H2: Hyperparameter optimisation significantly enhances the efficacy of CNN in detecting fraudulent transactions.

In the Experiment Design section of Chapter 3, we have discussed the six steps of the experiment with the aim of fraudulent transaction detection using Deep Learning. The steps are data collection, preprocessing, pre-analysis, splitting the test and training data, model training and optimisation, and model testing and evaluation. The CNN model, along with the learnable parameters and hyperparameters that influence the learning process, are discussed in detail in this chapter. We have explained how the model learns from data and can detect unseen transactions. Following that, we have provided definitions of various evaluation metrics.

6.5 Summary of Feature Engineering

In Chapter 4, we have integrated DFS, an explicit feature engineering technique with CNN, to improve fraudulent transaction detection performance. DFS operates on relational data by extracting features from the dataset based on the relationships among the data tables. These relationships encompass both forward connections, linking a single instance of one entity with a single instance of another entity, and backward connections, where a single instance of one entity is associated with multiple instances of another entity. Data extraction is based on three types of functions: (i) Entity Feature (EFEAT) representing the entity's own characteristics, (ii) Directed Feature (DFEAT) derived from the forward relationships with other entities, and (iii) Relation Feature (RFEAT) derived from the backward relationships within the dataset. Using our dataset as input, we have trained and deployed a CNN model for detecting fraudulent transactions, integrating DFS in the preprocessing stage to enhance the model’s performance. The use of DFS has

resulted in a notable enhancement of CNN’s performance, with a 10% improvement in precision and an 11% improvement in recall, F1 score, and accuracy.

In this chapter, our significant contribution lies in the incorporation of DFS with CNN, markedly elevating CNN's overall performance. Thus hypothesis H1, *DFS improves the fraudulent transaction detection performance of CNN significantly,* is validated.

6.6 Summary of Model Optimisation

In Chapter 5, we have focused on the critical aspect of hyperparameter tuning, which significantly influences model performance. This chapter has considered several pivotal hyperparameters, including the number of layers, number of filters, filter/kernel size, pool size, learning rate, activation function, dropout rate, batch size, and optimiser selection. We employed two sophisticated algorithms to optimise these hyperparameters: grid search and random search.

The outcomes of this hyperparameter optimisation are truly impressive. The performance of the optimised CNN models has been remarkable, with 7% enhancement across precision, recall, and F1 score for the model optimised with random search and 5.64%, 5.48% and 5.47% enhancement in terms of precision, recall, and F1 score for the model optimised with grid search when compared to the baseline model. Additionally, the CNN model optimised with random search exhibits an astounding 11% increase in AUC, and the CNN model optimised with grid search exhibits a remarkable 10% increase in AUC, demonstrating their enhanced performance in detecting fraudulent transactions.

Furthermore, the optimised CNN models surpass the baseline CNN and outperform a state-of-the-art method. To be specific, the optimised model employing random search displays a notable 10% improvement in recall rate, while the model utilising grid search demonstrates an impressive 8.9% increase in this performance metric. This substantiates the efficacy of our hyperparameter optimisation approach in bolstering the model's overall performance.

In this chapter, our significant contribution lies in the use of hyperparameter optimisation with CNN, significantly boosting CNN's overall efficacy. Hence hypothesis

H2, *Hyperparameter optimisation significantly enhances the efficacy of CNN in detecting fraudulent transactions*, is validated.

6.7 Limitations and Future Research Directions

While the study provided insight into fraud detection, the impact of features engineering on deep learning model, and optimisation of deep learning model, it is important to acknowledge the limitations, constraints, and future directions of the research.

1. Due to the scarcity of real data, open-source data was utilised, as financial institutions are reluctant to share their private data due to privacy and reputation concerns. However, the dataset used for the research is very realistic compared to natural data.

2. The fraud detection dataset is highly imbalanced, and although we used undersampling techniques were used to overcome the data imbalance problem, most data points from the majority class are lost and do not contribute to the model's training.

3. The proposed CNN model is a "black box" type model, meaning it is challenging to explain how and which features of the dataset contributed most to the model's ability to classify fraudulent transactions. This poses a challenge when addressing regulatory requirements for transparency and explainability.

4. The model has been developed using one dataset, and the model's performance may not be easily generalised to other domains or datasets.

5. The DFS feature engineering technique was utilised to improve the performance of CNN. However, there was no discrimination between more important and less important features. Developing new feature engineering techniques and employing attention mechanisms to assign

weights based on the importance of features would reduce training time while retaining superior performance.

6. Training and deploying deep learning models for fraud detection may require significant computational resources and time. Implementation in resource-constrained environments, such as real-time transaction processing systems, may be challenging. Additionally, high computational demands may limit the scalability of the model for large-scale applications.

7. The labelled data is used to train the supervised deep learning model, and the model does not work for data that is not labelled as fraudulent or non-fraudulent.

8. The deep learning model is trained on historical data. As fraudsters constantly change their tactics and fraud patterns, the model trained on the dataset may be degraded in novel kinds of fraudulent transactions.

9. Due to time constraints, limited availability of funds, and the scarcity of available fraudulent transactions, the developed model was tested with one dataset. However, testing the model with multiple datasets can be further explored.

10. In this project, hyperparameter optimisation techniques such as grid search and random search have been applied to attain the best performance of CNN. While these have yielded improved performance, further exploration of optimisation algorithms (like genetic algorithms and simulated annealing) and a broader search space could lead to even higher performance.

References

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., & Penn, G. (2012). Applying Convolutional Neural Networks Concepts to Hybrid NN-HMM Model for Speech Recognition. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4277-4280.

ABS. (2022). *Australian Bureau of Statistics 2021-22, .* https://www.abs.gov.au/statistics/people/crime-and-justice/personal-fraud/latest-release

Ahmed, M., Naser Mahmood, A., & Hu, J. (2016). A Survey of Network Anomaly Detection Techniques. *Journal of Network and Computer Applications*, *60*, 19-31. https://doi.org/https://doi.org/10.1016/j.jnca.2015.11.016

Akinsola, J. E. T. (2017). Supervised Machine Learning Algorithms: Classification and Comparison. *International Journal of Computer Trends and Technology (IJCTT)*, *48*, 128-138. https://doi.org/10.14445/22312803/IJCTT-V48P126

Alarfaj, F. K., Malik, I., Khan, H. U., Almusallam, N., Ramzan, M., & Ahmed, M. (2022). Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms. *IEEE Access*, *10*, 39700-39715. https://doi.org/10.1109/ACCESS.2022.3166891

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a Convolutional Neural Network. *2017 International Conference on Engineering and Technology (ICET)*, 1-6.

Alhaidari, F., Shaib, N. A., Alsafi, M., Alharbi, H., Alawami, M., Aljindan, R., Rahman, A.-u., & Zagrouba, R. (2022). ZeVigilante: Zetecting Zero-day Malware Using Machine Learning and Sandboxing Analysis Techniques. *Computational Intelligence and Neuroscience*, *2022*.

Alharbi, A., Alshammari, M., Okon, O. D., Alabrah, A., Rauf, H. T., Alyami, H., & Meraj, T. (2022). A Novel text2IMG Mechanism of Credit Card Fraud Detection: A Deep Learning Approach. *ELECTRONICS*, *11*(5). https://doi.org/10.3390/electronics11050756

Ali, A., Jayaraman, R., Azar, E., & Sleptchenko, A. (2022). A Systematic Assessment of Genetic Algorithm (GA) in Optimizing Machine Learning Model: A Case Study from Building Science. *2022 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 0384-0389.

Ali, H. (2012). An Analysis of Identity Theft: Motives, Related Frauds, Techniques and Prevention. *Journal of Law and Conflict Resolution*, *4*(1), 1-12.

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021a). Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions. *Journal of big data*, *8*(1), 53. https://doi.org/10.1186/s40537-021-00444-8

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021b). Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions. *Journal of big data*, *8*, 1-74.

AusPayNet. (2023). *2023 Australian Payment Fraud Report*. https://www.auspaynet.com.au/sites/default/files/2023-08/Fraud_Report_2023.pdf

Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017). Credit Card Fraud Detection using Machine Learning Techniques: A Comparative Analysis. *2017 International Conference on Computing Networking and Informatics (ICCNI)*, 1-9.

Azhan, M., & Meraj, S. (2020). Credit Card Fraud Detection using Machine Learning and Deep Learning Techniques. *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, 514-518. https://doi.org/10.1109/ICISS49785.2020.9316002

Babu, A. M., & Pratap, A. (2020). Credit Card Fraud Detection Using Deep Learning. *2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, 32-36. https://doi.org/10.1109/RAICS51191.2020.9332497

Baldi, P., Brunak, S., Chauvin, Y., Andersen, C. A., & Nielsen, H. (2000). Assessing the Accuracy of Prediction Algorithms for Classification: An Overview. *Bioinformatics*, *16*(5), 412-424.

Barto, A. G., & Dietterich, T. G. (2004). Reinforcement Learning and Its Relationship to Supervised Learning. *Handbook of Learning and Approximate Synamic Programming*, *10*, 9780470544785.

Becherer, N., Pecarina, J., Nykl, S., & Hopkinson, K. (2019). Improving Optimization of Convolutional Neural Networks through Parameter Fine-tuning. *Neural Computing and Applications*, *31*, 3469-3479.

Benchaji, I., Douzi, S., El Ouahidi, B., & Jaafari, J. (2021). Enhanced Credit Card Fraud Detection based on Attention Mechanism and LSTM Deep Model [Article]. *Journal of big data*, *8*(1), Article 151. https://doi.org/10.1186/s40537-021-00541-8

Bengio, Y., Goodfellow, I., & Courville, A. (2017a). Convolutional Neural Networks *MIT press Cambridge*, *1*, 296-297.

Bengio, Y., Goodfellow, I., & Courville, A. (2017b). Deep Learning. *MIT press Cambridge*, *1*, 9-10.

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-parameter Optimization. *Journal of Machine Learning Research 13*(2).

Biscione, V., & Bowers, J. (2020). Learning Translation Invariance in CNNs. *arXiv:2011.11757*.

Boutaher, N., Elomri, A., Abghour, N., Moussaid, K., & Rida, M. (2020). A Review of Credit Card Fraud Detection Using Machine Learning Techniques. *2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech)*, 1-5. https://doi.org/10.1109/CloudTech49835.2020.9365916

Breskuvienė, D., & Dzemyda, G. (2023). Imbalanced Data Classification Approach Based on Clustered Training Set. *Data Science in Applications*, 43-62.

Brey, P., & Søraker, J. H. (2009). Philosophy of Computing and Information Technology. In A. Meijers (Ed.), *Philosophy of Technology and Engineering Sciences* (pp. 1341-1407). North-Holland. https://doi.org/https://doi.org/10.1016/B978-0-444-51667-1.50051-3

Buscema, M. (1998). Back Propagation Neural Networks. *Substance use & misuse*, *33*(2), 233-270.

Carrasco, R. S. M., & Sicilia-Urbán, M. A. (2020). Evaluation of Deep Neural Networks for Reduction of Credit Card Fraud Alerts. *IEEE Access*, *8*, 186421-186432. https://doi.org/10.1109/ACCESS.2020.3026222

Chang, S., Wang, C., & Wang, C. (2022). Automated Feature Engineering for Fraud Prediction in Online Credit Loan Services. *2022 13th Asian Control Conference (ASCC)*, 738-743.

Chen, P. P.-S. (1976). The Entity-relationship Model—Toward a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, *1*(1), 9-36.

Dang, T. K., Tran, T. C., Tuan, L. M., & Tiep, M. V. (2021). Machine Learning Based on Resampling Approaches and Deep Reinforcement Learning for Credit Card Fraud Detection Systems. *Applied Sciences - BASEL*, *11*(21). https://doi.org/10.3390/app112110004

Dongare, A., Kharde, R., & Kachare, A. D. (2012). Introduction to Artificial Neural Network. *International Journal of Engineering and Innovative Technology (IJEIT)*, *2*(1), 189-194.
Feurer, M., & Hutter, F. (2019). Hyperparameter Optimization. *Automated Machine Learning: Methods, Systems, Challenges*, 3-33.
Galajit, K., Karnjana, J., Unoki, M., & Aimmanee, P. (2019). Semi-fragile Speech Watermarking based on Singular-spectrum Analysis with CNN-based Parameter Estimation for Tampering Detection. *APSIPA Transactions on Signal and Information Processing*, *8*, e11.
Grossi, E., & Buscema, M. (2007). Introduction to Artificial Neural Networks. *European Journal of Gastroenterology & Hepatology*, *19*(12), 1046-1054.
Gür, A. (2022). Deep Feature Synthesis for Accurate Breast Cancer Prediction. 2022 Medical Technologies Congress (TIPTEKNO),
Guresen, E., & Kayakutlu, G. (2011). Definition of Artificial Neural Networks with Comparison to Other Networks. *Procedia Computer Science*, *3*, 426-433.
Gyamfi, N. K., & Abdulai, J.-D. (2018). Bank Fraud Detection using Support Vector Machine. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 37-41.
Hanley, J. A., & McNeil, B. J. (1982). The Meaning and use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, *143*(1), 29-36.
Hilal, W., Gadsden, S. A., & Yawney, J. (2022). Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances.
Hinton, G. E. (2009). Deep Belief Networks. *Scholarpedia*, *4*(5), 5947.
Hu, K., Wang, J., Liu, Y., & Chen, D. (2019). Automatic Feature Engineering from very High Dimensional Event Logs using Deep Neural Networks. *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, 1-9.
Ikeda, C., Ouazzane, K., Yu, Q., & Hubenova, S. (2021). New Feature Engineering Framework for Deep Learning in Financial Fraud Detection. *International Journal of Advanced Computer Science and Applications*, *12*(12), 10-21. https://doi.org/10.14569/IJACSA.2021.0121202
Jha, S., & Westland, J. C. (2013). A Descriptive Study of Credit Card Fraud Pattern. *Global Business Review*, *14*(3), 373-384. https://doi.org/10.1177/0972150913494713
Kanter, J. M., & Veeramachaneni, K. (2015). Deep Feature Synthesis: Towards Automating Data Science Endeavors. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 1-10.
Kewei, X., Peng, B., Jiang, Y., & Lu, T. (2021). A Hybrid Deep Learning Model For Online Fraud Detection. *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, 431-434. https://doi.org/10.1109/ICCECE51280.2021.9342110
Khazane, A., Rider, J., Serpe, M., Gogoglou, A., Hines, K., Bruss, C. B., & Serpe, R. (2019). DeepTrax: Embedding Graphs of Financial Transactions [Conference Paper]. *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*, 126-133, Article 8999046. https://doi.org/10.1109/ICMLA.2019.00028
Khurana, U., Turaga, D., Samulowitz, H., & Parthasrathy, S. (2016). Cognito: Automated Feature Engineering for Supervised Learning. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 1304-1307.
Kim, K. G. (2016). Book Review: Deep Learning. *Healthcare Informatics Research*, *22*(4), 351-354.
Kim, S.-H., Geem, Z. W., & Han, G.-T. (2020). Hyperparameter Optimization Method based on Harmony Search Algorithm to Improve Performance of 1D CNN Human

Respiration Pattern Recognition System. *Sensors*, *20*(13), 3697. https://mdpi-res.com/d_attachment/sensors/sensors-20-03697/article_deploy/sensors-20-03697-v2.pdf?version=1594018114

Kitchenham, B. C. (2007). *Guidelines for Performing Systematic Literature Reviews in Software Engineering* (Keele University and Durham University Joint Report, Issue. https://citeseerx.ist.psu. edu/viewdoc/download?doi=10.1.1.117.471&rep=rep1&type=pdf

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: A Review of Classification and Combining Techniques. *Artificial Intelligence Review*, *26*, 159-190.

Kraemer-Mbula, E., Tang, P., & Rush, H. (2013). The Cybercrime Ecosystem: Online Innovation in the Shadows? *Technological Forecasting and Social Change*, *80*(3), 541-555.

Levi, M. (2009). *Financial Crime. In: Michael Tonry (ed.), Oxford Handbook of Crime and Public Policy*. Oxford University Press.

Li, Z. C., Liu, G. J., & Jiang, C. J. (2020). Deep Representation Learning With Full Center Loss for Credit Card Fraud Detection. *IEEE Transactions On Computational Social Systems*, *7*(2), 569-579. https://doi.org/10.1109/TCSS.2020.2970805

Liashchynskyi, P., & Liashchynskyi, P. (2019). Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. *arXiv preprint arXiv:1912.06059*.

Lim, K. S., Lee, L. H., & Sim, Y.-W. (2021). A Review of Machine Learning Algorithms for Fraud Detection in Credit Card Transaction. *International Journal of Computer Science & Network Security*, *21*(9), 31-40.

Liu, J., Gu, X., & Shang, C. (2020). Quantitative Detection of Financial Fraud Based on Deep Learning with Combination of E-Commerce Big Data. *Complexity*, *2020*. https://doi.org/10.1155/2020/6685888

Liu, X., Yan, K., Kara, L. B., & Nie, Z. (2021). CCFD-Net: A Novel Deep Learning Model for Credit Card Fraud Detection. *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, 9-16. https://doi.org/10.1109/IRI51335.2021.00008

Medsker, L. R., & Jain, L. (2001). Recurrent Neural Networks. *Design and Applications*, *5*(64-67), 2.

Misra, S., Thakur, S., Ghosh, M., & Saha, S. K. (2020). An Autoencoder Based Model for Detecting Fraudulent Credit Card Transaction. *Procedia Computer Science*, *167*, 254-262. https://doi.org/10.1016/j.procs.2020.03.219

Mohamad, N., Ahmad, N. B., Jawawi, D. N. A., & Hashim, S. Z. M. (2020). Feature Engineering for Predicting MOOC Performance. *IOP conference series. Materials Science and Engineering*, *884*(1), 12070. https://doi.org/10.1088/1757-899X/884/1/012070

Mohammed, K. K., Hassanien, A. E., & Afify, H. M. (2022). Classification of Ear Imagery Database using Bayesian Optimization based on CNN-LSTM Architecture. *Journal of Digital Imaging*, *35*(4), 947-961. https://link.springer.com/content/pdf/10.1007/s10278-022-00617-8.pdf

Mubalaike, A. M., & Adali, E. (2018). Deep Learning Approach for Intelligent Financial Fraud Detection System. *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, 598-603. https://doi.org/10.1109/UBMK.2018.8566574

Naby, A. A. E., Hemdan, E. E.-D., & El-Sayed, A. (2021). Deep Learning Approach for Credit Card Fraud Detection. *2021 International Conference on Electronic Engineering (ICEEM)*, 1-5. https://doi.org/10.1109/ICEEM52022.2021.9480639

Najadat, H., Altiti, O., Aqouleh, A. A., & Younes, M. (2020). Credit Card Fraud Detection Based on Machine and Deep Learning. *2020 11th International Conference on*

Information and Communication Systems (ICICS), 204-208. https://doi.org/10.1109/ICICS49469.2020.239524
Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep Learning Applications and Challenges in Big Data Analytics. *Journal of Big Data*, *2*(1), 1-21.
Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E. B., & Turaga, D. S. (2017). Learning Feature Engineering for Classification. *Ijcai*, *17*, 2529-2535.
Ng, A. (2011). Sparse Autoencoder. *CS294A Lecture Notes*, *72*(2011), 1-19.
O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. *arXiv pre-print server*. https://doi.org/arxiv:1511.08458
Pang, G., Shen, C., Cao, L., & Van Den Hengel, A. (2021). Deep Learning for Anomaly Detection: A Review. *ACM Computing Surveys*, *54*(2), 1-38. https://doi.org/10.1145/3439950
Pillai, T. R., Hashem, I. A. T., Brohi, S. N., Kaur, S., & Marjani, M. (2018). Credit Card Fraud Detection Using Deep Learning Technique. *2018 Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA)*, 1-6. https://doi.org/10.1109/ICACCAF.2018.8776797
Podgor, E. S. (1999). Criminal Fraud. *American University Law Review*, 729–768.
Pratuzaite, G., & Maknickiene, N. (2020). Investigation of Credit Cards Fraud Detection by using Deep Learning and Classification Algorithms. *11Th International Scientific Conference Business And Management 2020*, 389-396. https://doi.org/10.3846/bm.2020.558
Pulver, A., & Lyu, S. (2017). LSTM with Working Memory. *2017 International Joint Conference on Neural Networks (IJCNN)*, 845-851.
Pumsirirat, A., & Yan, L. (2018). Credit Card Fraud Detection using Deep Learning based on Auto-encoder and Restricted Boltzmann Machine. *International Journal of Advanced Computer Science and Applications*, *9*(1), 18-25. https://doi.org/10.14569/IJACSA.2018.090103
Raghavan, P., & Gayar, N. E. (2019). Fraud Detection using Machine Learning and Deep Learning. *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 334-339. https://doi.org/10.1109/ICCIKE47802.2019.9004231
Ramamoorti, S. W. O. (2007). Fraud: The Human Factor. *Financial Executive*, 53–55.
Rao, C.-c., & Li, R.-w. (2021). Research on Prediction Method on RUL of Motor of CNC Machine based on Deep Learning. *International Journal of Computing Science and Mathematics*, *14*(4), 338-346.
Sanober, S., Alam, I., Pande, S., Arslan, F., Rane, K. P., Singh, B. K., Khamparia, A., & Shabaz, M. (2021). An Enhanced Secure Deep Learning Algorithm for Fraud Detection in Wireless Communication. *Wireless communications and mobile computing*, *2021*. https://doi.org/10.1155/2021/6079582
Sharma, N., Sharma, R., & Jindal, N. (2021). Machine Learning and Deep Learning Applications-A Vision. *Global Transitions Proceedings*, *2*(1), 24-28.
Shenvi, P., Samant, N., Kumar, S., & Kulkarni, V. (2019). Credit Card Fraud Detection using Deep Learning. *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, 1-5. https://doi.org/10.1109/I2CT45611.2019.9033906
Shinde, Y., Chadha, A. S., & Shitole, A. (2021). Detecting Fraudulent Transactions Using Hybrid Fusion Techniques. *2021 3rd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE)*, 1-10.
Smiles, J. A., & Sasi Kumar, A. (2019). Synthetic Minority Oversampling and Smote Regularised Deep Autoencoders Neural Network Techniques for Fraud Prediction in Financial Payment Services. *International Journal of Innovative Technology and Exploring Engineering*, *8*(12), 3908-3915. https://doi.org/10.35940/ijitee.L3419.1081219

Smith, A. D. (2004). Cybercriminal Impacts on Online Business and Consumer Confidence. *Online Information Review*, *28*(3), 224-234.
Stojanović, B., Božić, J., Hofer-Schmitz, K., Nahrgang, K., Weber, A., Badii, A., Sundaram, M., Jordan, E., & Runevic, J. (2021). Follow the Trail: Machine Learning for Fraud Detection in Fintech Applications. *Sensors (Basel)*, *21*(5), 1594. https://doi.org/10.3390/s21051594
Tran, T. C., & Dang, T. K. (2021). Machine Learning for Prediction of Imbalanced Data: Credit Fraud Detection. *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 1-7. https://doi.org/10.1109/IMCOM51814.2021.9377352
Ullah, I., Rios, A., Gala, V., & McKeever, S. (2022). Explaining Deep Learning Models for Tabular Data using Layer-wise Relevance Propagation [Article]. *Applied Sciences (Switzerland)*, *12*(1), Article 136. https://doi.org/10.3390/app12010136
Wang, C., Dou, Y., Chen, M., Chen, J., Liu, Z., & Yu, P. S. (2021). Deep Fraud Detection on Non-attributed Graph [Conference Paper]. *Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021*, 5470-5473. https://doi.org/10.1109/BigData52589.2021.9672028
Wang, H. B., Zhou, C., Wu, J., Dang, W. Z., Zhu, X. Q., Wang, J. L., & Ieee. (2018). *Deep Structure Learning for Fraud Detection* 2018 International Conference Data Mining (ICDM), https://ieeexplore.ieee.org/document/8594881/
Wang, M., Ding, Z., & Pan, M. (2020). LbR: A New Regression Architecture for Automated Feature Engineering. *2020 International Conference on Data Mining Workshops (ICDMW)*, 432-439. https://doi.org/10.1109/ICDMW51313.2020.00066
Wedge, R., Kanter, J. M., Santiago Moral, R., Sergio Iglesias, P., & Veeramachaneni, K. (2017). Solving the "False Positives" Problem in Fraud Prediction.
Wedge, R., Kanter, J. M., Veeramachaneni, K., Rubio, S. M., & Perez, S. I. (2019). Solving the False Positives Problem in Fraud Prediction Using Automated Feature Engineering. 372-388. https://doi.org/10.1007/978-3-030-10997-4_23
West, J., & Bhattacharya, M. (2016). Intelligent Financial Fraud Detection: A Comprehensive Review. *Computers & Security*, *57*, 47-66.
Wu, X., Sahoo, D., & Hoi, S. C. (2020). Recent Advances in Deep Learning for Object Detection. *Neurocomputing*, *396*, 39-64.
Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S., & Jiang, C. (2018). Random Forest for Credit Card Fraud Detection. *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, 1-6.
Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional Neural Networks: An Overview and Application in Radiology. *Insights into Imaging*, *9*(4), 611-629. https://doi.org/10.1007/s13244-018-0639-9
Yang, J., Tang, Z., Guan, Z., Hua, W., Wei, M., Wang, C., & Gu, C. (2021). Automatic Feature Engineering-Based Optimization Method for Car Loan Fraud Detection. *Discrete Dynamics in Nature and Society*, *2021*, 1-10. https://doi.org/10.1155/2021/6077540
Yinze, Y., Tingxi LI. (2022). Deep Learning Techniques for Financial Fraud Detection. *14th International Conference on Computer Research and Development* 16-22. https://doi.org/10.1109/ICCRD54409.2022.9730314
Youssef, B., Bouchra, F., & Brahim, O. (2020). Rules Extraction and Deep Learning for e-Commerce Fraud Detection. *2020 6th IEEE Congress on Information Science and Technology (CiSt)*, 145-150. https://doi.org/10.1109/CiSt49399.2021.9357066
Zabinsky, Z. B. (2009). Random Search Algorithms. *Department of Industrial and Systems Engineering*.

Zhang, W., Yang, G., Lin, Y., Ji, C., & Gupta, M. M. (2018). On Definition of Deep Learning. *2018 World Automation Congress (WAC)*, 1-5.
Zhang, X., Chen, X., Yao, L., Ge, C., & Dong, M. (2019). Deep Neural Network Hyperparameter Optimization with Orthogonal Array Tuning. *Neural Information Processing: 26th International Conference, ICONIP 2019*, 287-295.
Zhang, Z., & Huang, S. (2020). Credit Card Fraud Detection via Deep Learning Method Using Data Balance Tools. *2020 International Conference on Computer Science and Management Technology (ICCSMT)*, 133-137. https://doi.org/10.1109/ICCSMT51754.2020.00033

Appendix-A

Features generated by Deep Feature Synthesis

SL	Generated Features	SL	Generated Features
1	transactions_id	61	cards.MIN(transactions.MCC)
2	User	62	cards.MIN(transactions.Merchant City)
3	Card	63	cards.MIN(transactions.Merchant Name)
4	Year	64	cards.MIN(transactions.Merchant State)
5	Month	65	cards.MIN(transactions.Minutes)
6	Day	66	cards.MIN(transactions.Month)
7	Amount	67	cards.MIN(transactions.Use Chip)
8	Use Chip	68	cards.MIN(transactions.Year)
9	Merchant Name	69	cards.MIN(transactions.Zip)
10	Merchant City	70	cards.SKEW(transactions.Amount)
11	Merchant State	71	cards.SKEW(transactions.Card)
12	Zip	72	cards.SKEW(transactions.Day)
13	MCC	73	cards.SKEW(transactions.Errors?)
14	Errors?	74	cards.SKEW(transactions.Hour)
15	Hour	75	cards.SKEW(transactions.MCC)
16	Minutes	76	cards.SKEW(transactions.Merchant City)
17	cards.CARD INDEX	77	cards.SKEW(transactions.Merchant Name)
18	cards.Card Brand	78	cards.SKEW(transactions.Merchant State)
19	cards.Card Type	79	cards.SKEW(transactions.Minutes)
20	cards.Card Number	80	cards.SKEW(transactions.Month)
21	cards.CVV	81	cards.SKEW(transactions.Use Chip)
22	cards.Has Chip	82	cards.SKEW(transactions.Year)
23	cards.Cards Issued	83	cards.SKEW(transactions.Zip)
24	cards.Credit Limit	84	cards.STD(transactions.Amount)
25	cards.Year PIN last Changed	85	cards.STD(transactions.Card)
26	cards.Card on Dark Web	86	cards.STD(transactions.Day)
27	cards.COUNT(transactions)	87	cards.STD(transactions.Errors?)
28	cards.MAX(transactions.Amount)	88	cards.STD(transactions.Hour)
29	cards.MAX(transactions.Card)	89	cards.STD(transactions.MCC)
30	cards.MAX(transactions.Day)	90	cards.STD(transactions.Merchant City)
31	cards.MAX(transactions.Errors?)	91	cards.STD(transactions.Merchant Name)
32	cards.MAX(transactions.Hour)	92	cards.STD(transactions.Merchant State)
33	cards.MAX(transactions.MCC)	93	cards.STD(transactions.Minutes)
34	cards.MAX(transactions.Merchant City)	94	cards.STD(transactions.Month)
35	cards.MAX(transactions.Merchant Name)	95	cards.STD(transactions.Use Chip)
36	cards.MAX(transactions.Merchant State)	96	cards.STD(transactions.Year)
37	cards.MAX(transactions.Minutes)	97	cards.STD(transactions.Zip)

38	cards.MAX(transactions.Month)	98	cards.SUM(transactions.Amount)
39	cards.MAX(transactions.Use Chip)	99	cards.SUM(transactions.Card)
40	cards.MAX(transactions.Year)	100	cards.SUM(transactions.Day)
41	cards.MAX(transactions.Zip)	101	cards.SUM(transactions.Errors?)
42	cards.MEAN(transactions.Amount)	102	cards.SUM(transactions.Hour)
43	cards.MEAN(transactions.Card)	103	cards.SUM(transactions.MCC)
44	cards.MEAN(transactions.Day)	104	cards.SUM(transactions.Merchant City)
45	cards.MEAN(transactions.Errors?)	105	cards.SUM(transactions.Merchant Name)
46	cards.MEAN(transactions.Hour)	106	cards.SUM(transactions.Merchant State)
47	cards.MEAN(transactions.MCC)	107	cards.SUM(transactions.Minutes)
48	cards.MEAN(transactions.Merchant City)	108	cards.SUM(transactions.Month)
49	cards.MEAN(transactions.Merchant Name)	109	cards.SUM(transactions.Use Chip)
50	cards.MEAN(transactions.Merchant State)	110	cards.SUM(transactions.Year)
51	cards.MEAN(transactions.Minutes)	111	cards.SUM(transactions.Zip)
52	cards.MEAN(transactions.Month)	112	cards.DAY(Acct Open Date)
53	cards.MEAN(transactions.Use Chip)	113	cards.DAY(Expires)
54	cards.MEAN(transactions.Year)	114	cards.MONTH(Acct Open Date)
55	cards.MEAN(transactions.Zip)	115	cards.MONTH(Expires)
56	cards.MIN(transactions.Amount)	116	cards.WEEKDAY(Acct Open Date)
57	cards.MIN(transactions.Card)	117	cards.WEEKDAY(Expires)
58	cards.MIN(transactions.Day)	118	cards.YEAR(Acct Open Date)
59	cards.MIN(transactions.Errors?)	119	cards.YEAR(Expires)
60	cards.MIN(transactions.Hour)		