PROJECT REPORT ON

DEEP LEARNING WORKSHOP WITH PYTHON (CSE3194)

Fraud Detection System using Neural Networks

Submitted in partial fulfillment of the
requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

Submitted by:

SUBHASISH NAHAK	**REG. NO: 2241014003**
ANSUJIT DALEI	**REG. NO: 2241004008**
SMARAN SATAPATHY	**REG. NO: 2241019342**

Center for Artificial Intelligence & Machine Learning
Department of Computer Science and Engineering
Institute of Technical Education and Research
Sikhsa 'O' Anusandhan
(Deemed to be University)
Bhubaneswar

May, 2025

Declaration

We hereby declare that the project report titled **"Fraud Detection System using Neural Networks "** is our own work, carried out under the guidance of **Jagseer Singh Sir**. We have not plagiarized any content and have duly cited all references.

ANSUJIT DALEI

SUBHASISH NAHAK

SMARAN SATAPATHY

Acknowledgement

We, the students of **B.Tech** in **Computer Science and Engineering (AI & ML)**, hereby acknowledge that we take full responsibility for the content, information, results, and findings presented in this project titled **"Fraud Detection System Using Neural Networks,"** submitted to **Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar**, in partial fulfillment of the requirements for the course **Deep Learning Workshop with Python (CSE 3194).**

We affirm that all efforts have been made to respect intellectual property rights, and proper credit has been given to the contributions, ideas, and work of others wherever applicable. This project has been prepared with academic integrity, and all referenced sources have been duly acknowledged for educational purposes. Furthermore, we declare that in the event of any unintentional infringement or violation of copyright or intellectual property rights, we, as the project authors, take complete responsibility for the same.

We would like to express our heartfelt gratitude to our project guide, **Jagseer Singh Sir** for their invaluable support, guidance, and feedback throughout the duration of this work. Their encouragement and insights were essential in shaping the outcome of this project.

We also extend our sincere thanks to the **Department of Computer Science and Engineering (AI & ML)** and the faculty members for providing us with the academic environment and resources necessary to carry out this project.

We are grateful to **Siksha 'O' Anusandhan (Deemed to be University)** for giving us the opportunity and platform to explore this area of deep learning and for their overall support in conducting the workshop.

Table of Contents

Contents

Abstract

In today's digital economy, financial transactions are increasingly vulnerable to fraudulent activities, posing serious risks to both individuals and organizations. Forgery of credit card transactions are seen one a frequent basis. But traditional fraud detection systems based on static rules and threshold monitoring often fail to detect complex and evolving fraud patterns. To address this challenge, this project presents a fraud detection system powered by neural networks, utilizing the capabilities of deep learning to analyse transaction data with important transactional features and identify suspicious behaviour with high accuracy.

The system is trained on labelled datasets containing both legitimate and fraudulent transactions. Using a multi-layered neural network, the model learns hidden patterns, relationships, and anomalies that typically signify fraudulent activity. Various performance metrics such as accuracy, precision, recall, area under curve and F1-score are used to evaluate the effectiveness of the model. Techniques like data normalization and hybrid of oversampling and undersampling are applied to address issues like class imbalance and improve detection accuracy.

The proposed approach demonstrates improved fraud detection capability to traditional methods. This project showcases how neural networks can be a powerful tool in developing intelligent, automated, and reliable fraud detection systems for real-world applications.

Keywords:

Fraud Detection, Neural Networks, Deep Learning, Financial Transactions, Machine Learning, Imbalanced Dataset, Synthetic Minority Oversampling Technique ,Random UnderSampling, Stochastic Gradient Descent Optimizer, Adam Optimizer, Root Mean Square Propagation Optimizer.

Chapter 1 – Introduction

1.1. Background and motivation

In recent years, digital payments like online banking and e-wallets have become more common, but they also bring more risk of fraud. Old fraud detection methods, like rule-based systems, can't keep up with how fast fraud techniques change. They often miss real fraud or flag too many false alerts. This project aims to solve that by using neural networks, a type of artificial intelligence that learns from data and finds hidden patterns. Our goal is to build a smart fraud detection system that gets better over time and helps keep online financial transactions safe and trustworthy.

1.2. Problem statement

Today, online financial fraud is a big problem for people, companies, and banks. Traditional fraud detection systems use fixed rules, which don't work well anymore. These systems often miss clever fraud tricks or wrongly flag safe transactions, causing money loss and frustration.

One major issue is that fraud cases are rare, so the data is unbalanced. This makes it hard for models to learn properly. We need a system that can handle this challenge.

This project uses neural networks to build a smart fraud detection system. Neural networks can learn patterns from past transactions and detect new fraud quickly and accurately, even with imbalanced data.

1.3. Objectives

The main goal of this project is to build a smart and reliable fraud detection system using neural networks. This system should be able to spot fake financial transactions quickly and accurately.

Here are the key steps we aim to achieve:

1. Understand how fraud happens – We will study real transaction data to learn the behaviour and unusual patterns of fraud.
2. Build a neural network model – We will create a layered model that can learn from complex data and tell the difference between normal and fraud transactions.
3. Fix unbalanced data – Since fraud cases are rare, we'll use techniques like undersampling, oversampling (SMOTE), normalization, and scaling to make the data more balanced.
4. Test how well the model works – We'll measure the model's accuracy and reliability using tools like precision, recall, F1-score, and ROC-AUC.

1.4. Scope

This project focuses on building and testing a fraud detection system using neural networks. The goal is to find fake or suspicious financial transactions by studying past data and spotting

unusual patterns. We will use deep learning techniques to train the system on pre-labelled datasets, where each transaction is already marked as fraud or not.

The system will use data preprocessing methods to fix issues like unbalanced data, where there are very few fraud cases compared to normal ones. We will test the model using different accuracy measures to make sure it works well.

Right now, the project uses offline, structured datasets and assumes the data is clean and properly labelled. While the current design is not set up for real-time use, it can be adjusted for that in the future. Features like real-time monitoring or adding this system into real banking software are outside the current project but could be added later.

1.5. Organisation of the Report

The Project Report has been classified into the following chapters:

Chapter 1: Introduction

It provides a brief overview of the project like the background, motivation, problem statement, objectives, and scope.

Chapter 2: Literature Review

It contains the details on the list of research papers of past 5 years using various methodologies for fraud detection system.

Chapter 3: System Design and Methodology

It contains information on dataset, preprocessing steps, working of the model and various algorithms used.

Chapter 4: Implementation Details

It describes the programming language used and details of the code.

Chapter 5: Results and Discussion

It describes the evaluation metrics and results obtained.

Chapter 6: Conclusion and Future Work

This chapter comprises of the limitations found and what are the future aspects.

References

Lists all research papers, articles, tools, and datasets referred to during the course of the project.

Appendices

Includes additional materials such as code snippets and screenshots.

Chapter 2 – Literature Review

2.1. Existing methods and models

The given list of research papers provide us the insight of various methodologies used in Fraud Detection System.

1) Credit Card Fraud Detection: **A Realistic Modeling and a Novel Learning Strategy *by Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi*** published in year **2017**.

The research paper present a fraud detection model using machine learning under realistic conditions. They address class imbalance with SMOTE, reduce features using PCA, and test models like Logistic Regression, Random Forest, and XGBoost. **XGBoost** outperforms due to better precision, recall, and **ROC-AUC** metrics. It also emphasizes handling concept drift and verification latency for improved fraud detection. [1]

2) Fraud Detection using Machine Learning By *Oladimeji Kazeem Python Programming IU University of Applied Sciences* published in the year **2023.**

The research paper explores fraud detection via real-time systems and ML models. After preprocessing and **EDA**, models such as **SVM**, **Random Forest**, and **K-Means** are applied. **PCA** and feature engineering improve performance. Random Forest showed highest accuracy. Model integration through APIs highlights deployment concerns like scalability and security. Evaluation metrics include F1-score, precision, and ROC curve. [2]

3) Financial Fraud Detection Based on Machine Learning: A Systematic Literature Review by *Abdulalem Ali* , *Shukor Abd Razak, Siti Hajar Othman ,Arafat Al-Dhaqm , Maged Nasser , Taiseer Abdalla, Elfadil Eisa , Tusneem Elhassan, Hashim Elshafie* and *Abdu Saif* published in year **2022.**

The research paper conducts a systematic review of 93 studies on ML in fraud detection using the **Kitchenham SLR method**. Techniques include **SVM**, **ANN**, **Random Forest**, **HMM**, and ensemble methods. Challenges like *data imbalance* and *concept drift* are discussed. The review encourages using deep learning and word embeddings such as *Word2Vec* and *BERT* to enhance detection accuracy. [3]

4) Credit Card Fraud Detection with Deep Learning Techniques by Ismael *K. Tonui* and *Preston Kibet* published in the year **2025.**

propose a deep learning-based fraud detection system using **CNN**, **LSTM**, and **Autoencoders**. Preprocessing includes **SMOTE** and **PCA**. **XGBoost** achieves the highest F1-score and accuracy, while deep models show strong potential. Each model handles complex patterns differently—CNN detects local features, LSTM captures sequences, and Autoencoders aid in anomaly detection.[4]

5) Credit Card Fraud Detection: **A Deep Learning Approach** by *Sourav Verma* and *Joydip Dhar* published in the year **2024.**

This research paper uses a hybrid deep learning approach combining feedforward networks and autoencoders. **SMOTE** and sampling techniques address class

imbalance. Feature optimization is performed using **Binary Bat Algorithm**. The model outperforms conventional classifiers like **SVM** and **Logistic Regression**, achieving **96.21% AUC**, demonstrating the effectiveness of deep learning with nature-inspired optimization. [5]

6) Credit Card Fraud Detection Using Machine Learning Credit Card Fraud Detection Using Machine Learning by *Meera AlEmad* published in the year **2022.**

The paper proposes compares four supervised models—**SVM**, **KNN**, **Naïve Bayes**, and **Logistic Regression**—using the CRISP-DM framework. SVM achieves the best performance with **99.94% accuracy**. Minimal preprocessing and balanced evaluation using confusion matrix metrics validate results. The study recommends using additional features like location data and trying more datasets and algorithms in the future. [6]

7) A novel method for detecting credit card fraud problems by **HaiChao Du , Li Lv,Hongliang Wang, An Guo** published in the year ***2024.***

The research paper proposes AE-XGB-SMOTE-CGAN, a hybrid deep learning model for imbalanced fraud datasets. **AutoEncoder** extracts features, **XGBoost** handles classification, and CGAN refines synthetic samples. This dual-phase augmentation boosts minority class representation. The model achieves **89.3% TPR** and **0.8845 MCC**, outperforming **KNN** and Random Forest, and proving its effectiveness in realistic fraud detection [7]

8) A Research Paper on Credit Card Fraud Detection by **Bora Mehar Sri Satya Teja1, Boomi Reddy Munendra, Mr. S. Gokulkrishnan** published in the year ***2022.***

The paper evaluates Random Forest and Decision Tree classifiers for detecting credit card fraud in an imbalanced dataset using transaction data from European cardholders. After applying SMOTE to address class imbalance, Random Forest outperformed with an accuracy of 99.98%, recall of 100%, and F1-score of 99.98%. The study confirms Random Forest's superior performance in fraud detection when combined with oversampling techniques. [8]

9) **Credit Fraud || Dealing with Imbalanced Datasets** by **Janio Martinez Bachmann** published in the recent years**.**

This paper explores various predictive models to detect fraudulent transactions in an imbalanced dataset. The author addresses class imbalance by creating a balanced subset with a **50/50** ratio of fraud and non-fraud cases. Multiple classifiers are evaluated to assess their accuracy in distinguishing between normal and fraudulent transactions. This practical guide offers insights into handling imbalanced datasets in fraud detection. [9]

10) **Credit Card Fraud Detection Using Improved Deep Learning Models** by **SumayaS. Sulaiman, IbraheemNadher** and **Sarab M. Hameed** published in the year ***2024***.

This paper focuses on developing a deep learning framework using **AE**, **CNN**, and **LSTM** with automated hyperparameter tuning. **SMOTE**, **ADASYN**, and RUS improve class balance. Bayesian optimization and binary focal loss enhance model focus on rare

fraud cases. LSTM with SMOTE achieves top results with **99.2% accuracy** and **96.3% AUC**, *outperforming traditional benchmarks*. [10]

2.2. Comparison with your approach

Aspect	Literature Findings	Our Approach
Data Imbalance Handling	Emphasizes addressing class imbalance via undersampling, oversampling (e.g., SMOTE). Undersampling may lose information; oversampling may overfit.	Uses a hybrid approach: applies SMOTE (oversampling) followed by RandomUnderSampler(undersampling).
Feature Scaling / Normalization	Normalizing features improves model performance and training convergence.	Applies StandardScaler for feature normalization. Aligns with best practices.
Model Selection	Neural networks, SVMs, and decision trees are commonly used. Ensemble methods (Random Forest, XGBoost) often show superior performance.	Implemented a simple neural network with two hidden layers (64 and 32 units). Uses dropout for regularization, experimented with activation function like ReLU, tanh and optimizers like SGD, Adam, RMS propagation for better understanding and smoother operation on the dataset.
Evaluation Metrics	Accuracy is insufficient due to imbalance. Emphasis on precision, recall, F1-score, AUC.	Evaluates using accuracy, precision, recall, F1-score, AUC a robust and recommended approach.

Chapter 3 – System Design and Methodology

3.1. Dataset description

The credit card fraud detection dataset consists of **284,807 transactions** collected over two days in September 2013 by European cardholders. It is **highly imbalanced**, with only **0.172%** labeled as **fraud**, making it suitable for anomaly detection research. The dataset includes 30 features: 28 anonymized variables **(V1–V28)** generated using **Principal Component Analysis (PCA)** to protect sensitive data, along with two original features — Time, representing the seconds since the first transaction, and Amount, indicating the transaction value. The **target variable**, Class, denotes whether a transaction is **fraud (1) or legitimate (0)**. This dataset supports supervised learning approaches.

3.2. Exploratory Data Analysis

We started our project with **Exploratory Data Analysis (EDA)** to understand the dataset better. Using **df.head()** and **df.info(),** we checked the structure, data types, and found no missing values. With df.describe(), we looked at the basic statistics like mean and range of the features. We also used **df['Class'].value_counts()** to see how many transactions were fraud and non-fraud. It showed a huge imbalance—very few fraud cases. To fix this, we planned to apply techniques like oversampling or hybrid methods to balance the data, so our model can learn to detect fraud more accurately.

3.3. Preprocessing steps

To ensure effective model training, we followed a structured preprocessing pipeline on the credit card fraud dataset. The first step was separating the features (X) from the target variable (y). Features included transaction details, while the target indicated whether the transaction was fraudulent (1) or not (0). Next, we performed feature normalization using StandardScaler, which transformed all features to have a mean of 0 and standard deviation of 1. This was important because the data contained variables on different scales, and normalization prevents larger-scale features from dominating the learning process.

A major challenge in fraud detection is class imbalance, as fraudulent transactions are extremely rare. To tackle this, we used a hybrid approach combining SMOTE and RandomUnderSampler. SMOTE creates synthetic samples for the minority class (fraud), while RandomUnderSampler reduces the majority class (non-fraud) to create a more balanced dataset. This approach ensured that our model was trained on a dataset that fairly represented both classes. Together, these preprocessing steps formed the foundation for building a fair and reliable fraud detection model.

3.4. Model architecture

The proposed model is an Artificial Neural Network (ANN) designed to detect credit card fraud using a structured deep learning approach. The network begins with an input layer that receives 30 numerical features derived from transaction data, including PCA-transformed components. This input is then passed through two hidden layers, each designed to capture complex patterns within the data. The first hidden layer consists of 64 neurons, followed by a second hidden layer with 32 neurons. Activation functions such as ReLU and Tanh are applied to introduce non-linearity, allowing the model to learn intricate relationships.

To improve generalization and reduce overfitting, dropout regularization with a dropout rate of 0.5 is applied after each hidden layer. The output layer comprises a single neuron with a sigmoid activation function, which outputs a probability representing the likelihood of a transaction being fraudulent.

The model is compiled using the binary cross-entropy loss function, suitable for binary classification problems. Various optimizers ing Adam, SGD, and RMSprop are used to evaluate performance, with metrics such as accuracy, precision, recall, F1-score, and AUC serving as evaluation criteria. This architecture aims to achieve robust and balanced fraud detection, even in the presence of class imbalance and data complexity.

3.5. Description of the Algorithms

Some of the important algorithms used in the model are:

(i) Normalization

Normalization is a crucial data preprocessing technique employed to scale numerical features to a uniform range, typically between 0 and 1. This ensures that all input features contribute equally to the model's learning process, preventing dominance by features with larger magnitudes. **Normalization** improves the convergence speed and performance of gradient-based optimization algorithms and is especially important for distance-based algorithms like k-NN and clustering models. By transforming the data into a consistent scale, **normalization** facilitates efficient training of machine learning models and enhances overall predictive accuracy.

(ii) SMOTE (Synthetic Minority Oversampling Technique)

SMOTE is an advanced oversampling technique used to address the issue of class imbalance in datasets, which often hinders model performance. Instead of merely replicating minority class instances, **SMOTE** generates synthetic examples by interpolating between existing ones. This is achieved by selecting a random minority class sample and one of its nearest neighbors, then creating a new sample along the line segment connecting them. For instance, in fraud detection, where fraudulent transactions may form only 10% of the dataset, **SMOTE** helps balance the classes by synthetically augmenting the fraud cases, leading to better learning by the model.

(iii) Random UnderSampling (RUS)

Random UnderSampling (RUS) is a technique used to correct class imbalance by reducing the number of instances in the majority class. Unlike **SMOTE**, which adds synthetic data to the minority class, **RUS** works by randomly discarding samples from the overrepresented

class to balance the class distribution. This approach helps reduce training time and alleviates model bias toward the majority class. In fraud detection, where legitimate transactions dominate, **RUS** removes a subset of these legitimate entries to ensure the model pays adequate attention to the minority (fraudulent) class, thereby improving detection performance.

(iv) Artificial Neural Network (ANN)

An **Artificial Neural Network (ANN)** is a machine learning model inspired by the functioning of the human brain. It comprises layers of interconnected processing units called neurons that work collectively to recognize complex patterns in data. **ANNs** are especially effective in tasks involving classification and prediction. In our fraud detection model, we utilized *ReLU* and *tanh* as activation functions across different layers, coupled with optimizers such as *Stochastic Gradient Descent*, *Adam*, and *Root Mean Square Propagation* to minimize the loss function effectively. The **ANN** learns to distinguish fraudulent from legitimate transactions by iteratively adjusting the weights based on errors during training.

Chapter 4 – Implementation Details

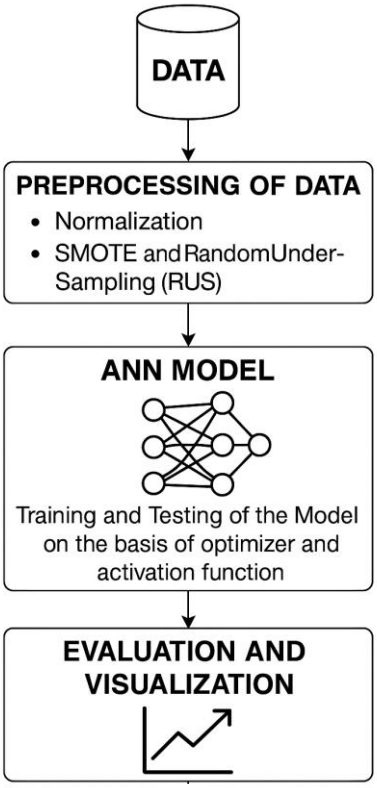
4.1. Programming languages, frameworks

This machine learning project leverages **Python**, a popular language for AI and machine learning due to its simplicity and wide range of libraries. **TensorFlow** and **Keras**, central to the system, enable the building and training of deep learning models. **Keras**, as a high-level API on TensorFlow, simplifies neural network construction. Data manipulation is handled by **Pandas**, while **NumPy** aids in efficient numerical operations. For data analysis and visualization, **Matplotlib** and **Seaborn** provide insights through graphs and charts. The project is executed in **Google Colab**, benefiting from free GPU access. To handle imbalanced data, **SMOTE** and **Random UnderSampling (RUS)** are applied to balance the dataset, improving model performance.

4.2. Code modules description

Category	Modules/Packages Used	Purpose
Data Handling	pandas, numpy	Load, manipulate, and perform numerical operations on tabular data
Visualization	matplotlib.pyplot, seaborn	Plot data and results, with enhanced statistical and aesthetic features
Preprocessing	train_test_split, StandardScaler (from sklearn), SMOTE (from imblearn), RUS(Random UnderSampling)	Split data, scale features, and balance classes using oversampling and undersampling.
Evaluation Metrics	confusion_matrix, classification_report, precision_score, recall_score, etc.	Evaluate model performance using classification metrics
Deep Learning	tensorflow, keras.models, keras.layers, keras.optimizers, callbacks	Build, compile, train, and tune deep learning models

4.3. System Working



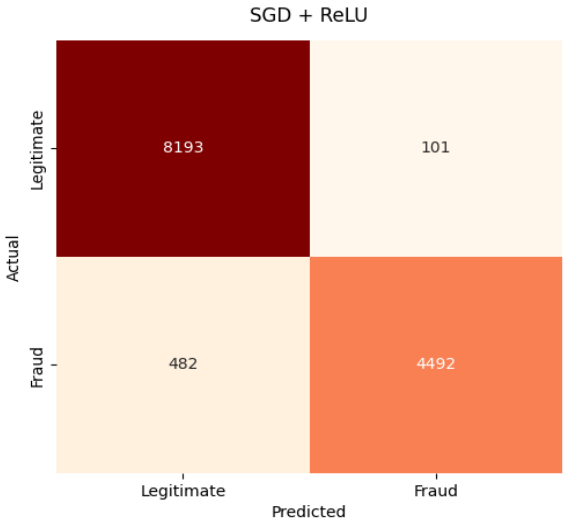
Category	Undersampling	Oversampling	Hybrid Sampling
What it does	Reduces the number of majority class samples.	Increases the number of minority class samples by duplicating or synthetically generating them (e.g., SMOTE).	Combines both oversampling and undersampling techniques.
Pros	- Fast training (smaller dataset). - Reduces risk of overfitting to the majority class.	- Preserves all majority class data. - Often works well with small datasets.	- Balances classes more effectively. - Reduces noise introduced by synthetic samples and redundancy in majority class.
Cons	- Risk of losing important information from the majority class. - Not suitable for very small datasets.	- Can lead to overfitting, especially if duplicating data. - Increases training time (larger dataset).	- More complex to implement. - Longer preprocessing time.

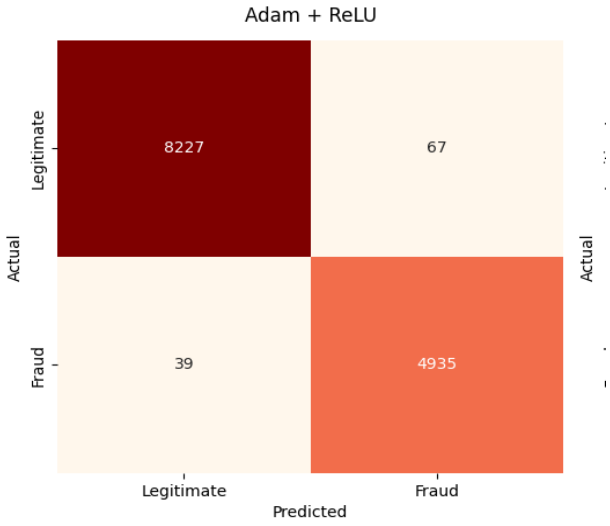
According to our requirements, we have chosen to use **Hybrid methods.**

Then our model approaches towards the Artificial Neural Network, where by the use of various optimizers and activation functions, our model is trained and performance metrics are monitored accordingly.

Chapter 5 – Results and Discussion

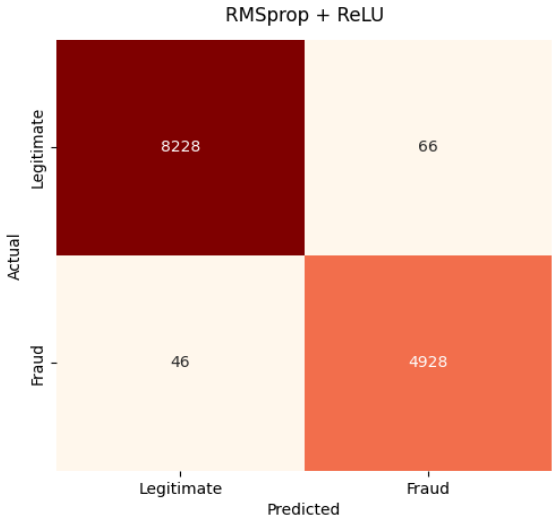
5.1. Evaluation metrics

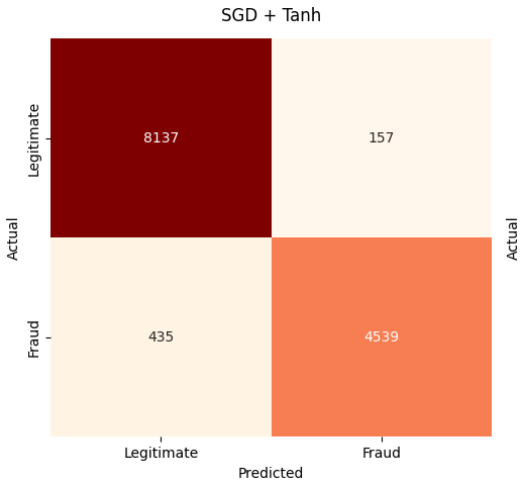


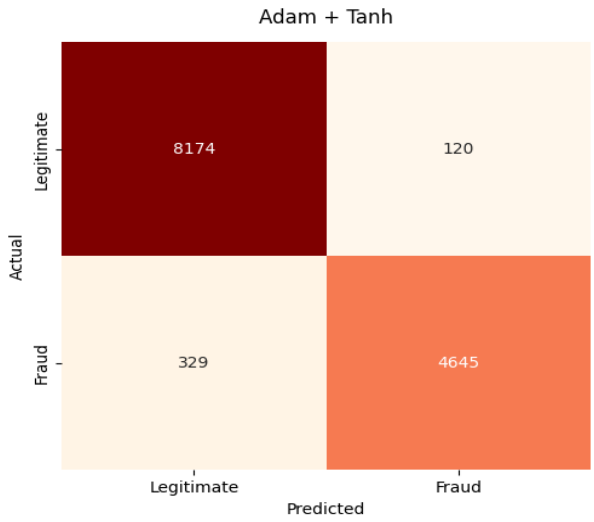


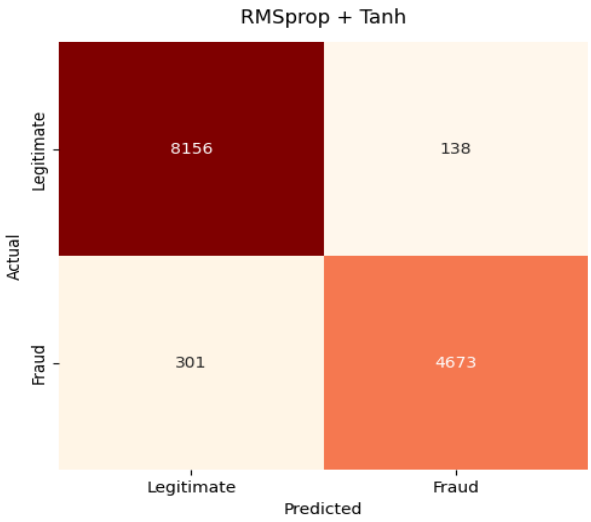
SGD + Tanh

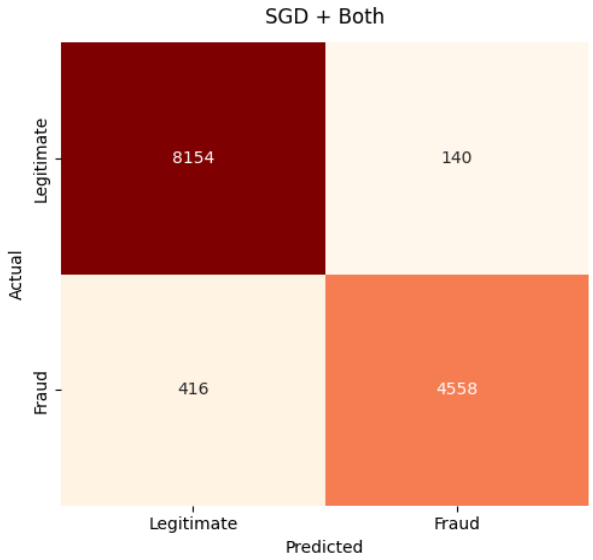
Adam + Tanh

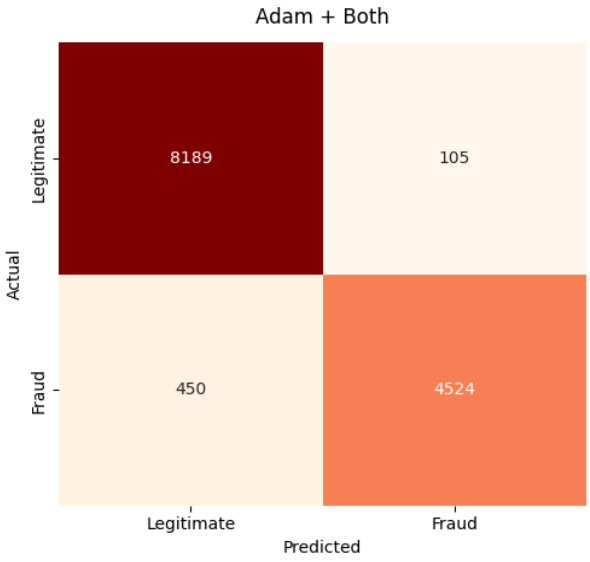


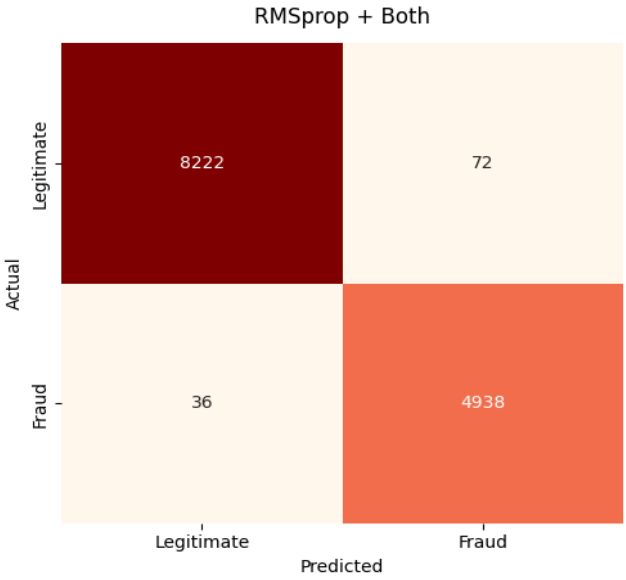












5.2. Results

Classification Reports:

```
SGD + ReLU:
Accuracy: 0.9575, Precision: 0.9814, Recall: 0.9037, AUC: 0.9949, F1_score: 0.9947
Classification Report:
              precision    recall  f1-score   support

           0     0.9449    0.9898    0.9668      8294
           1     0.9814    0.9037    0.9410      4974

    accuracy                         0.9575     13268
   macro avg     0.9632    0.9467    0.9539     13268
weighted avg     0.9586    0.9575    0.9571     13268
```

```
Adam + ReLU:
Accuracy: 0.9958, Precision: 0.9906, Recall: 0.9982, AUC: 0.9998, F1_score: 0.9998
Classification Report:
              precision    recall  f1-score   support

           0     0.9989    0.9943    0.9966      8294
           1     0.9906    0.9982    0.9944      4974

    accuracy                         0.9958     13268
   macro avg     0.9948    0.9963    0.9955     13268
weighted avg     0.9958    0.9958    0.9958     13268
```

```
RMSprop + ReLU:
Accuracy: 0.9952, Precision: 0.9902, Recall: 0.9970, AUC: 0.9997, F1_score: 0.9996
Classification Report:
              precision    recall  f1-score   support

           0     0.9982    0.9941    0.9961      8294
           1     0.9902    0.9970    0.9936      4974

    accuracy                         0.9952     13268
   macro avg     0.9942    0.9955    0.9949     13268
weighted avg     0.9952    0.9952    0.9952     13268
```

```
SGD + Tanh:
Accuracy: 0.9575, Precision: 0.9634, Recall: 0.9216, AUC: 0.9912, F1_score: 0.9911
Classification Report:
              precision    recall  f1-score   support

           0     0.9542    0.9790    0.9664      8294
           1     0.9634    0.9216    0.9420      4974

    accuracy                         0.9575     13268
   macro avg     0.9588    0.9503    0.9542     13268
weighted avg     0.9576    0.9575    0.9573     13268
```

```
Adam + Tanh:
Accuracy: 0.9747, Precision: 0.9746, Recall: 0.9574, AUC: 0.9972, F1_score: 0.9970
Classification Report:
              precision    recall  f1-score   support

           0     0.9747    0.9850    0.9799      8294
           1     0.9746    0.9574    0.9659      4974

    accuracy                         0.9747     13268
   macro avg     0.9747    0.9712    0.9729     13268
weighted avg     0.9747    0.9747    0.9746     13268
```

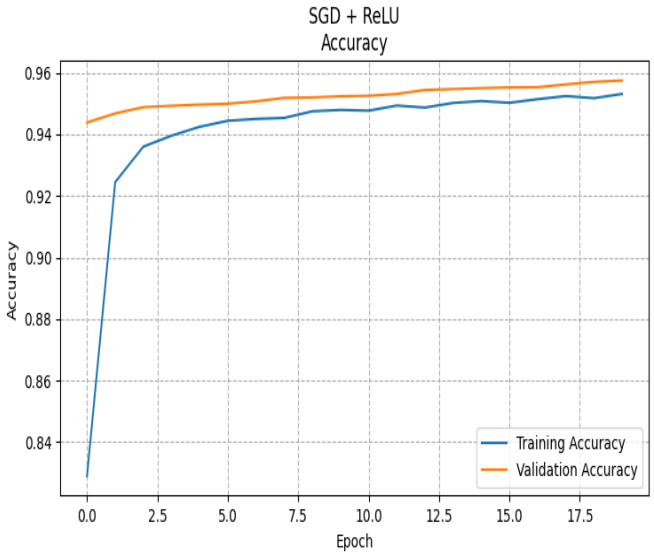
```
RMSprop + Tanh:
Accuracy: 0.9741, Precision: 0.9765, Recall: 0.9540, AUC: 0.9967, F1_score: 0.9964
Classification Report:
              precision    recall  f1-score   support

           0     0.9728    0.9863    0.9795      8294
           1     0.9765    0.9540    0.9651      4974

    accuracy                         0.9741     13268
   macro avg     0.9747    0.9701    0.9723     13268
weighted avg     0.9742    0.9741    0.9741     13268
```

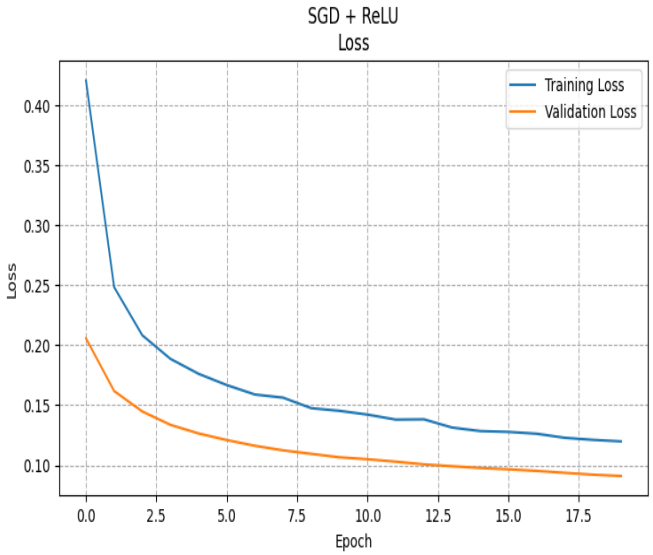
```
SGD + Both:
Accuracy: 0.9677, Precision: 0.9783, Recall: 0.9345, AUC: 0.9960, F1_score: 0.9957
Classification Report:
              precision    recall  f1-score   support

           0     0.9617    0.9876    0.9745      8294
           1     0.9783    0.9345    0.9559      4974

    accuracy                         0.9677     13268
   macro avg     0.9700    0.9610    0.9652     13268
weighted avg     0.9679    0.9677    0.9675     13268
```

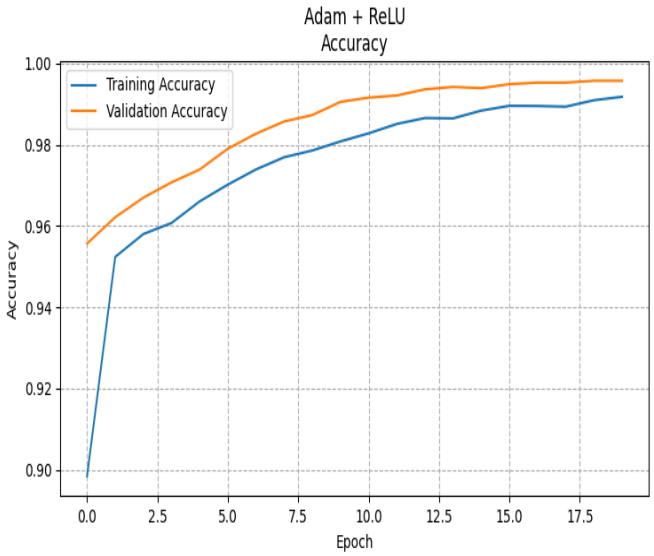
```
Adam + Both:
Accuracy: 0.9641, Precision: 0.9771, Recall: 0.9260, AUC: 0.9957, F1_score: 0.9956
Classification Report:
              precision    recall  f1-score   support

           0     0.9570    0.9870    0.9717      8294
           1     0.9771    0.9260    0.9509      4974

    accuracy                         0.9641     13268
   macro avg     0.9670    0.9565    0.9613     13268
weighted avg     0.9645    0.9641    0.9639     13268
```

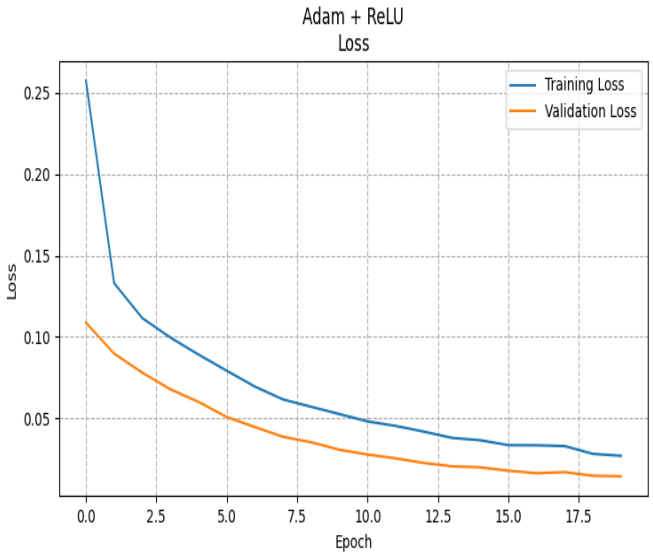
```
RMSprop + Both:
Accuracy: 0.9954, Precision: 0.9914, Recall: 0.9964, AUC: 0.9997, F1_score: 0.9991
Classification Report:
              precision    recall  f1-score   support

           0     0.9978    0.9948    0.9963      8294
           1     0.9914    0.9964    0.9939      4974

    accuracy                         0.9954     13268
   macro avg     0.9946    0.9956    0.9951     13268
weighted avg     0.9954    0.9954    0.9954     13268
```

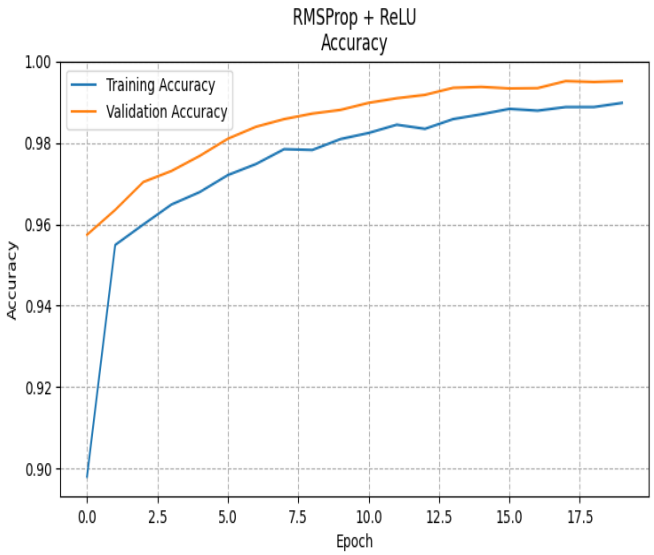
Graph of Accuracy and Loss Comparisons for some of the optimizers and activation functions:

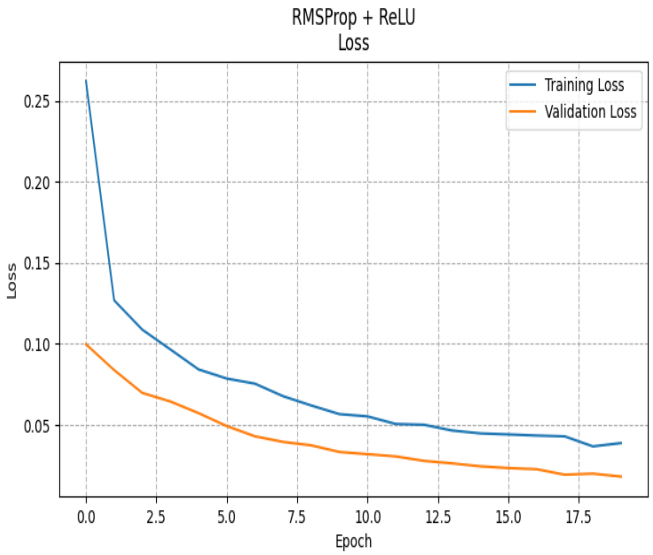


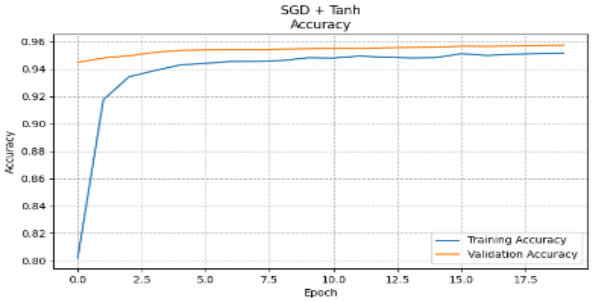


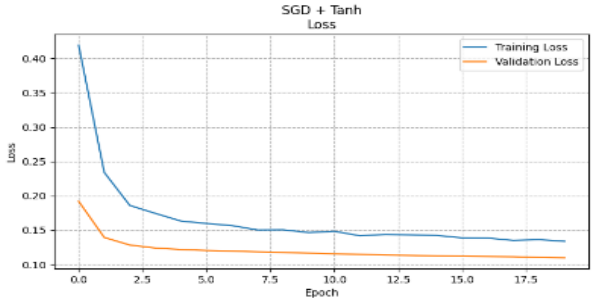


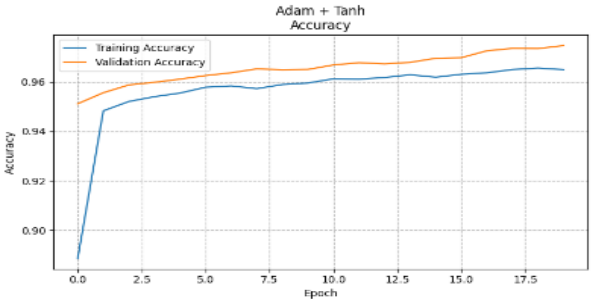


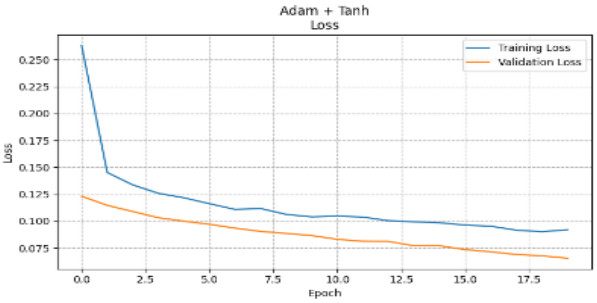


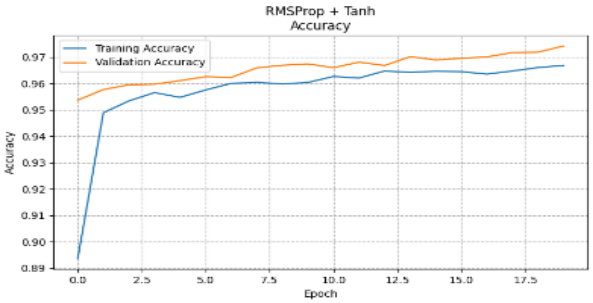


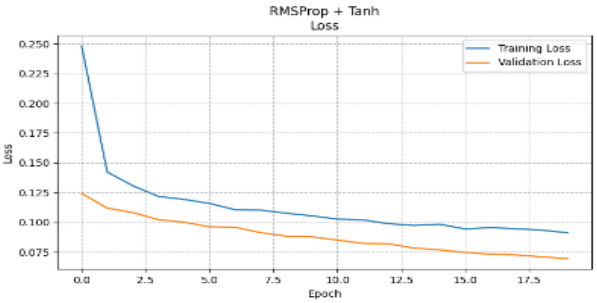

SGD + Tanh
Accuracy
Training Accuracy
Validation Accuracy
Epoch

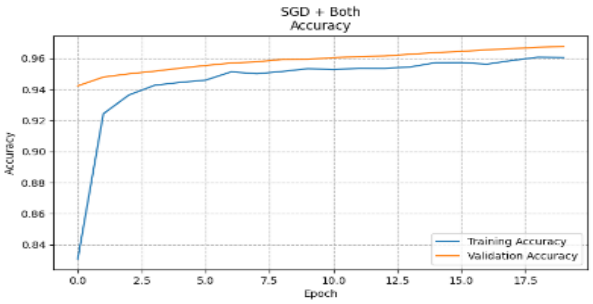

SGD + Tanh
Loss
Training Loss
Validation Loss
Epoch

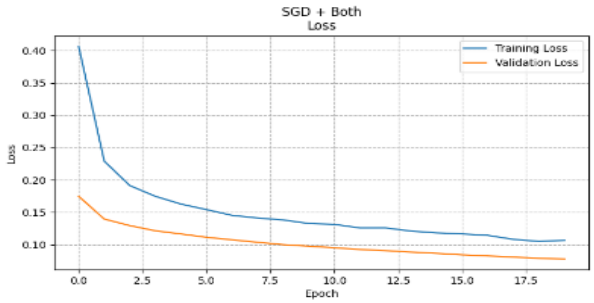

Adam + Tanh
Accuracy
Training Accuracy
Validation Accuracy
Epoch

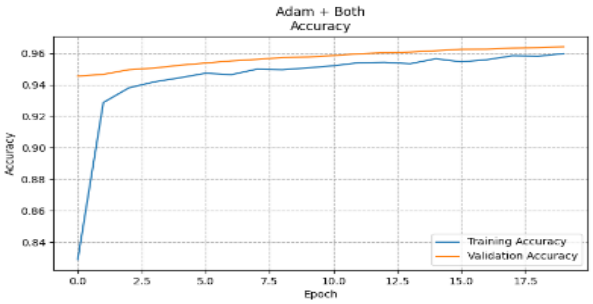

Adam + Tanh
Loss
Training Loss
Validation Loss
Epoch

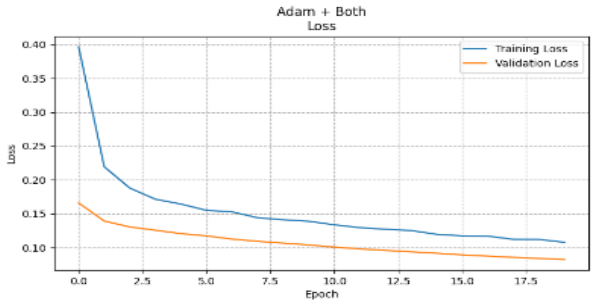

RMSProp + Tanh
Accuracy
Training Accuracy
Validation Accuracy
Epoch

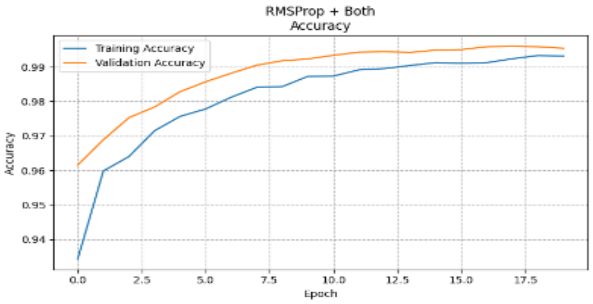

RMSProp + Tanh
Loss
Training Loss
Validation Loss
Epoch

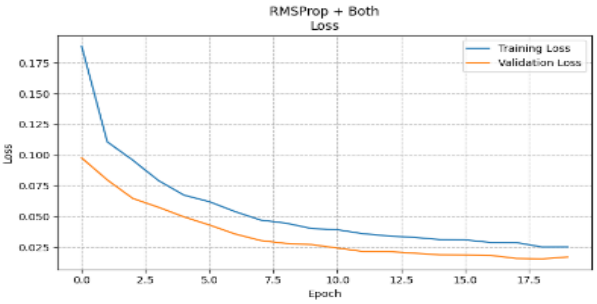

SGD + Both
Accuracy
Training Accuracy
Validation Accuracy
Epoch


SGD + Both
Loss
Training Loss
Validation Loss
Epoch


Adam + Both
Accuracy
Training Accuracy
Validation Accuracy
Epoch

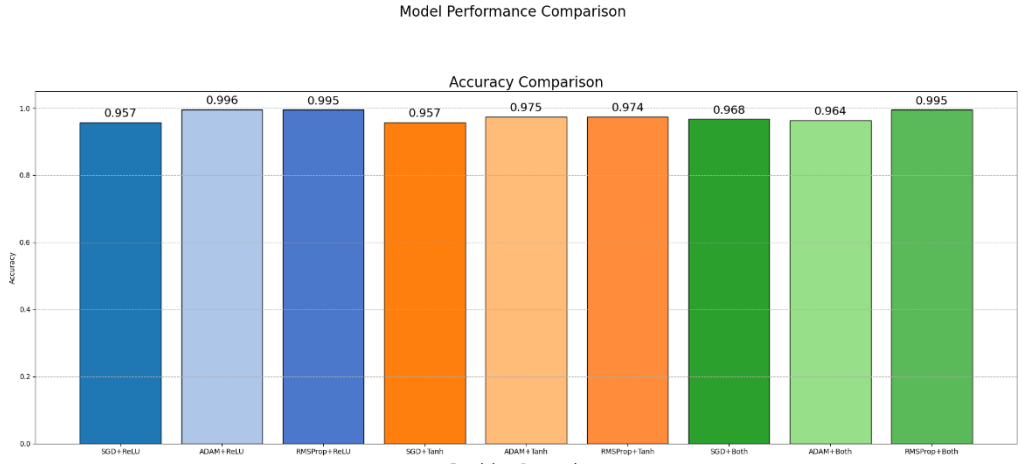

Adam + Both
Loss
Training Loss
Validation Loss
Epoch

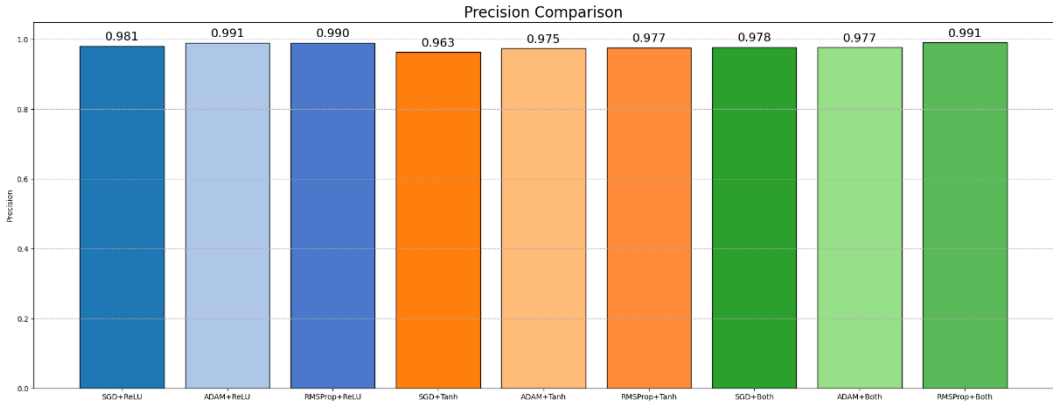

RMSProp + Both
Accuracy
Training Accuracy
Validation Accuracy
Epoch

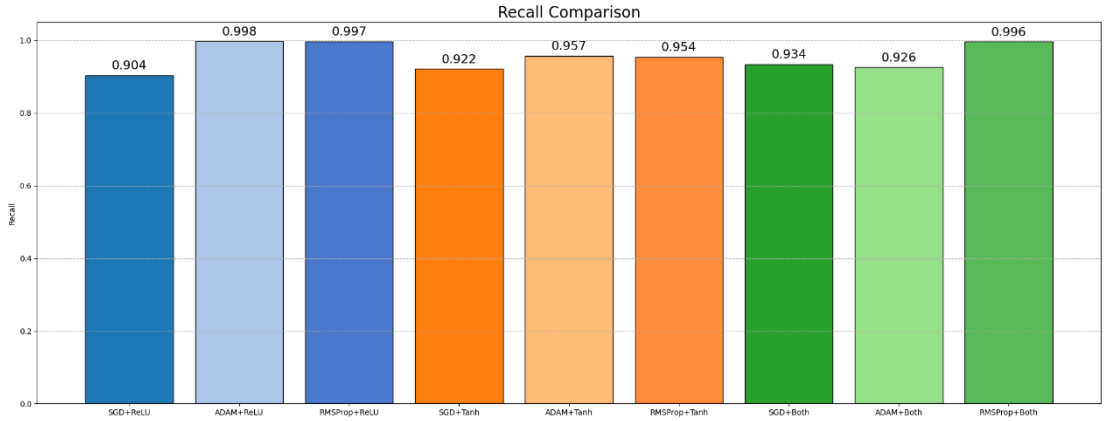

RMSProp + Both
Loss
Training Loss
Validation Loss
Epoch

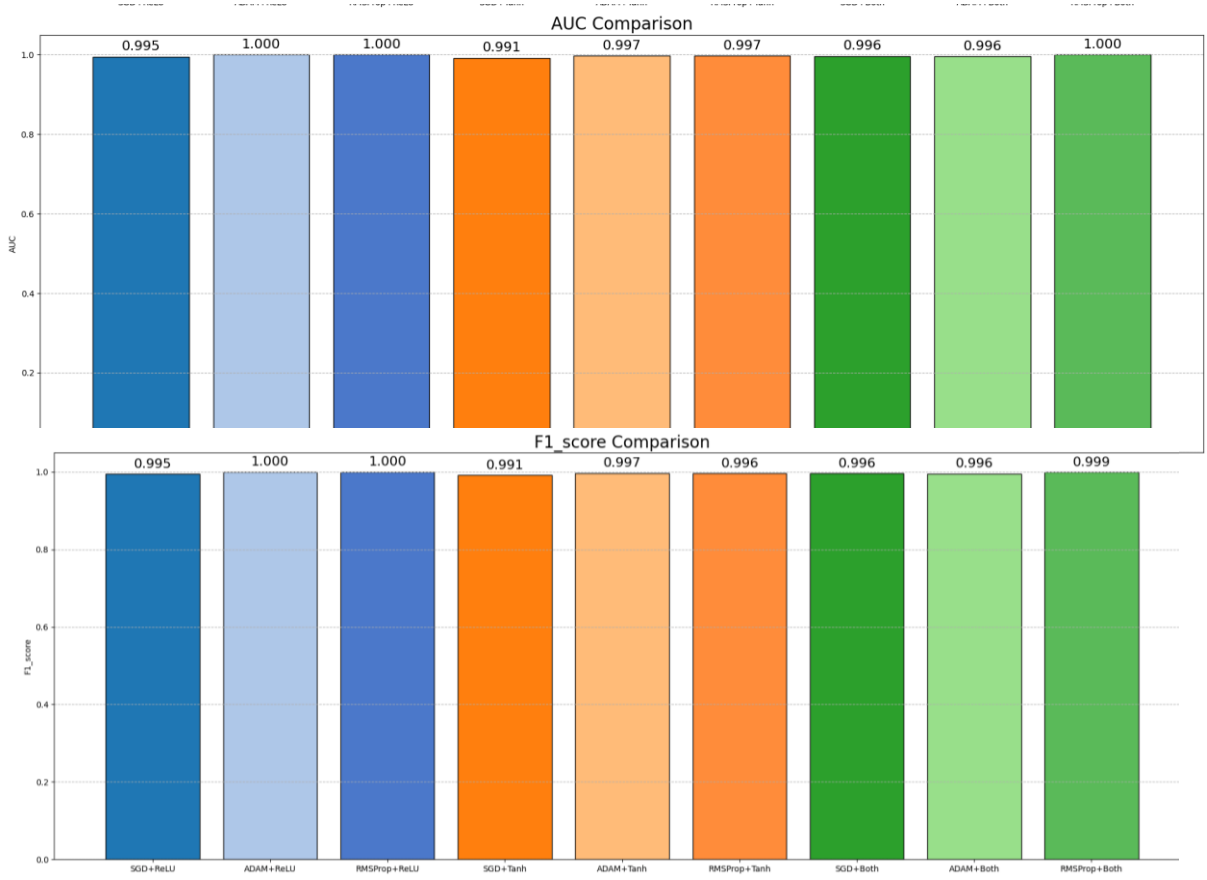
Graph on the basis of Performance Metrics:

Sequence of graph : (SGD+ReLU,Adam+ReLU,RMSProp+ReLU,SGD+tanh,Adam+tanh,RMSProp+tanh SGD+both,Adam+both,RMSProp+both)

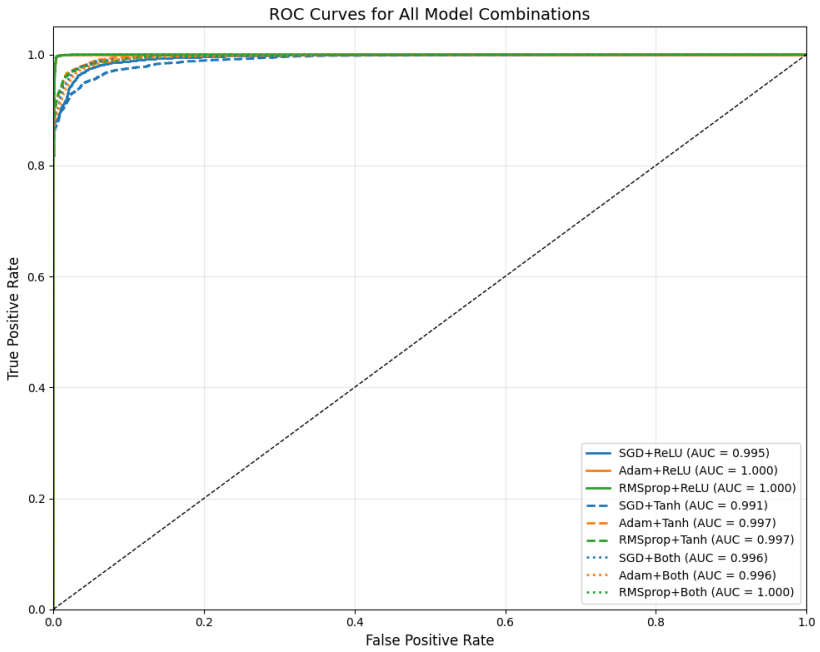







AUC Comparison
0.995
1.000
1.000
0.991
0.997
0.997
0.996
0.996
1.000
AUC
F1_score Comparison
0.995
1.000
1.000
0.991
0.997
0.996
0.996
0.996
0.999
F1_score
SGD+ReLU
ADAM+ReLU
RMSProp+ReLU
SGD+Tanh
ADAM+Tanh
RMSProp+Tanh
SGD+Both
ADAM+Both
RMSProp+Both

ROC Curve:


ROC Curves for All Model Combinations
True Positive Rate
False Positive Rate
SGD+ReLU (AUC = 0.995)
Adam+ReLU (AUC = 1.000)
RMSprop+ReLU (AUC = 1.000)
SGD+Tanh (AUC = 0.991)
Adam+Tanh (AUC = 0.997)
RMSprop+Tanh (AUC = 0.997)
SGD+Both (AUC = 0.996)
Adam+Both (AUC = 0.996)
RMSprop+Both (AUC = 1.000)

5.3. Discussion

After balancing the data using hybrid methods, we have obtained a data consisting approximately **62.5% majority class** and **37.5% minority class**, which was preferable for our ANN model. Then when the dataset was passed through ANN model, we got **confusion matrix**, **performance metrices** and **ROC curve.** We observed this for all the mentioned optimizers and activation functions and were inferred about the accuracy of the model.

Chapter 6 – Conclusion & Future Work

6.1. Summary of outcomes

(i)We conducted **Exploratory Data Analysis** and displayed majority and minority class of the imbalanced data and the balanced data using **Pie charts (hybrid methods** were used for *balancing* the data)**.**

(ii) ANN was used consisting of different optimizers and activation functions. Then according to the model we obtained training and validation accuracy and loss. We obtained **confusion matrix**, **classification report** and **graphs** were implemented accordingly.

6.2. Limitations

Some of the limitations faced by our model is:

(i) It lacks real-time functionality.

(ii) Due to relying upon a single model prototype, there is potential vulnerability to adversarial attacks.

(iii) Absence of continuous learning as new fraud techniques arise day to day, so it may be incompatible/inefficient to handle them.

6.3. Future improvements

Some of the future improvements which can be done are:

(i) Implement the model for real-time fraud detection by integrating it with stream-processing frameworks like Apache Kafka or Apache Flink.

(ii) Exploring deeper and more complex architectures like Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) models, and autoencoders to enhance the model's ability to capture temporal patterns and detect anomalies.

(iii) Optimize model deployment for cloud platforms like AWS or Google Cloud Platform and lightweight edge devices to ensure scalability, portability, and efficient performance in real-world applications.

(iv) Combining the neural network with rule-based systems or other models like Bayesian networks to leverage the strengths of different approaches and enhance flexibility in handling diverse fraud scenarios.

References

1. J. R. Vyas, *Fraud Detection Book*, VVP Engineering College, Rajkot, n.d. [1]

2. O. Kazeem, *Fraud Detection Using Machine Learning*, University of Stirling, 2025. [2]

3. A. Ali *et al.*, "Financial fraud detection based on machine learning: A systematic literature review," *Applied Sciences*, vol. 12, no. 19, p. 9637, 2022. [3]

4. I. Tonui and P. Kibet, *Credit Card Fraud Detection with Deep Learning Techniques*, Austin Peay State University, 2025. [4]

5. S. Verma and J. Dhar, *Credit Card Fraud Detection: A Deep Learning Approach*, ABV-IIITM Gwalior, n.d. [5]

6. M. AlEmad, *Credit Card Fraud Detection Using Machine Learning*, 2022. [6]

7. H. Du, L. Lv, H. Wang, and A. Guo, "A novel method for detecting credit card fraud problems," 2024. [7]

8. B. M. Sri Satya Teja, B. R. Munendra, and S. Gokulkrishnan, "A research paper on credit card fraud detection," 2022. [8]

9. J. M. Bachmann, *Credit Fraud: Dealing with Imbalanced Datasets*, via Kaggle[9]

10. S. S. Sulaiman, I. Nadher, and S. M. Hameed, "Credit card fraud detection using improved deep learning models," 2024. [10]

11. www.google.com.[11]

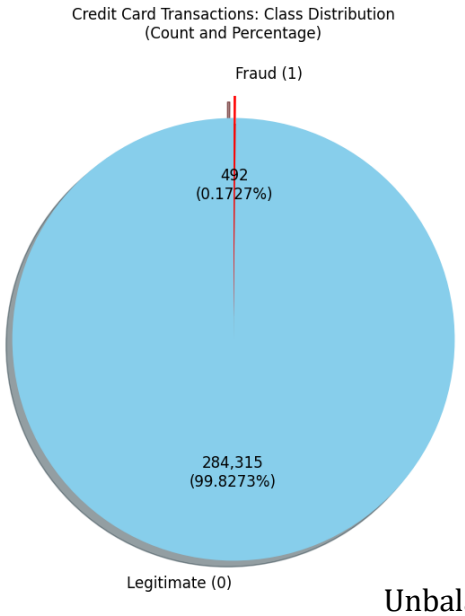
Appendices

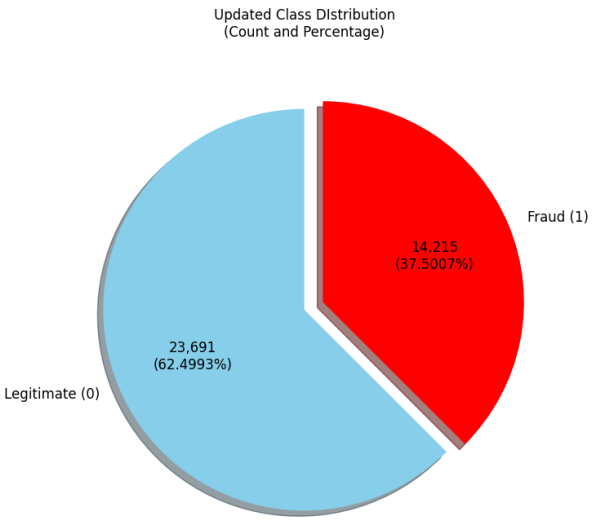
Additional code snippets

```
# Normalize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
smote = SMOTE(sampling_strategy=0.05,random_state=42)
Xn_resampled, yn_resampled = smote.fit_resample(X_scaled, y)
rus = RandomUnderSampler(sampling_strategy=0.6, random_state=42)
X_resampled, y_resampled = rus.fit_resample(Xn_resampled, yn_resampled)
```

```
mport tensorflow as tf
rom tensorflow.keras.models import Sequential
rom tensorflow.keras.layers import Dense,Dropout
rom sklearn.model_selection import train_test_split
rom matplotlib import pyplot as plt
mport numpy as np

=X_resampled
=y_resampled

 y=tf.keras.utils.to_categorical(y,num_classes=2)
_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.35,random_state=42)
 x_train,x_test=x_train/255.0,x_test/255.0
ef create_model(optimizer, hidden_activation):
    model = Sequential([
        Dense(64, input_dim=X.shape[1], activation=hidden_activation),
        Dropout(0.5),
        Dense(32, activation=hidden_activation),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])

    model.compile(optimizer=optimizer,
                  loss='binary_crossentropy',
                  metrics=['accuracy',
                           tf.keras.metrics.Precision(name='precision'),
                           tf.keras.metrics.Recall(name='recall'),
                           tf.keras.metrics.AUC(name='auc'),
                           tf.keras.metrics.F1Score(name='f1')])
    return model
```

Screenshots or logs





Unbalanced and Balanced Data