Credit Card Fraud Detection Using Convolutional Neural Networks

Hasan Ahmati[1]

Other group members:
Leonardo Koxhaj[2], **Ergi Mira**[3], **Eugene Acheampong**[4]

Abstract. Banking transactions, such as credit card transactions are gaining popularity quickly. At the same time frauds are also increasing, causing costumers and the financial institutions to lose billions of dollars annually. There have been several methodologies used to detect and prevent credit card fraud, but with no significant success. In the recent years, multiple studies have used machine learning techniques as means to solve this issue. In this paper, we present a study of convolutional neural network (CNN) used on a given dataset and a comparison between different studies already made for this model. The proposed model has achieved 0,858 accuracy, 0,96 precision and 0,7475 recall.

1 Introduction

In the financial ecosystem, the quest to make the life easier led to the continuous and rapid transformation of the world through digital technology. People have moved from exchanging goods for goods in the barter system of trade to transacting using cash, and recently through credit card and online payment services [4]. However, various type of credit card fraud has resulted in loss of huge amount of money. According to [12], over £1.2 billion were stolen through fraud in 2022, from which 78% of the cases start online.

Over the years, to evolve with the evolution of fraud detection methods, attackers have also changed and improved their fraud practices to avoid detection. Therefore, techniques to detect and prevent credit card fraud must constantly be updated. In our approach, we use the CNN algorithm, which is designed to detect fraud using different layers that detect patterns in a transaction and then evaluate either that transaction is fraud or not.

Some reasons why the CNN algorithm is used in this study:

- Even why CNN is primary designed for image recognition task, in our scenario it is important and can be applied to analyse sequential patterns [1].
- The weight sharing feature, which reduces the number of trainable network parameters and helps the network to enhance generalization and to avoid overfitting [1].
- The data representation of credit card transaction might be suited better for neural network architectures such as CNN.

This study focuses on the application of the CNNs to a dataset that contains a total of 1 296 675 credit card transactions. It makes sure that there's a diverse range of scenarios. Furthermore, it contains a wide scope of features, such as: transaction date and time, the merchant, category, location, transaction time etc.

In the following sections, we will cover the background of the study, discussing about different papers already made in this field. Subsequently, we present the experiments tried and the results obtained and analyse the results. Finally, the research will conclude by summarizing the key findings and results and outline possible opportunities for future work.

2 Abbreviations

LC- Learning Curve; CNN- Convolutional Neural Network; BN- Batch Normalization; CL- Convolutional Layer; CC-credit card, CCFD-Credit card fraud detection

3 Background

In recent years, with the CC fraud increasing and with the traditional approaches that used rule-based methods and conventional anomaly detection failing as they are time-consuming, resource-intensive, and inaccurate [2], the eyes of the CC industries turned towards the already exciting new field of machine learning as they were becoming more and more popular. This way a lot of big financial companies that deal with CCs decided to develop deep learning algorithms to analyse and detect patterns that can prevent CC fraud. In this paragraph, we will present different works, related to CNN and neural network techniques

CNN, even why is mostly used for image recognition tasks, is one of the deep learning algorithms that it used for CCFD. Various studies have been proposed in this area. [3] proposed a CCFD model using CNN as the classifier and Synthetic Minority Oversampling Technique (SMOTE) to balance the dataset. Using three evaluation metrics, the performance of the model was compared with that of a Multi-Layer Perceptron model which was trained using both balanced and unbalanced datasets. [10] does a comparison of 3 neural networks: Simple Neural Network (SNN), Multilayer Perception Layer (MPL) and CNN and comes to conclusion that the MPL proved to have the best accuracy with their dataset, with 87.88%.

[1] School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK, email: ha2116d@gre.ac.uk
[2] School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK, email: lk9921w@gre.ac.uk
[3] School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK, email: em6495p@gre.ac.uk
[4] School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK, email: ea6728s@gre.ac.uk

While CNN comes second with an accuracy of 82,86%. [4] does a comparison of a two CNN models, one balanced dataset with ADASYN and the other imbalanced. They conclude that the model using imbalanced dataset has a slightly better accuracy, while the model using ADASYN gets much better results in precision, recall and f1. While [11], does a comparison between four deep learning models ANN, CNN, LSTM, RNN and tries different balancing techniques, from which Random oversampling results to be the best to balance their dataset. This study proposes the use of LSTM as the best approach, while CNN is ranked 3rd with 0,82 precision, 0,78 recall and 0,801 f1 score.

As most of these papers did not include their code in their report, started searching for different reports published on GitHub with their code included. [11] had included their code there but realised that they have used a lot of external methods so decided to take effective parts from this report and code, adapt the metrics of [4],balance the dataset and take the architecture from [8].

4 Experiments and results

4.1 Dataset

This study is based on a dataset that firstly contains 1296674 samples, from which 7506 are fraudulent, and 24 features that contain transaction and client information. Dataset can be found in this link: https://www.kaggle.com/datasets/kartik2112/fraud-detection

4.2 Dataset balancing

An important process of our implementation is the dataset balancing. A pie chart helped us to check the transaction class distribution.

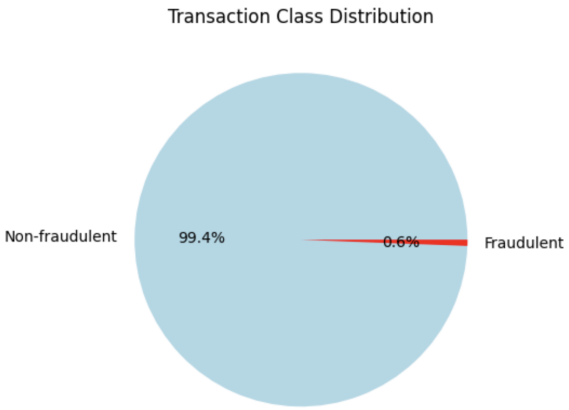

Figure 1. Dataset's class distribution

From the pie chart we can see a major class imbalance, where only 0.6% of the transactions are fraudulent. Class imbalance can lead towards a biased model, misleading of the evaluation metrics and an increase of the false negatives. To address this imbalance, different balancing techniques could help. As mentioned in the Background section in [4] ADASYN gave exceptional results with almost max number of results in all the metrics. Oversampling was not to convincing because it can lead to overfitting as it duplicates instances of the minority class. Also, this could bring a higher computational cost during training as the dataset size increases. The technique used to balance the dataset is by matching the non-fraudulent instances with the fraudulent ones with the removal of all the extra instances in the majority class. The class balance after using this technique is represented in figure 2:

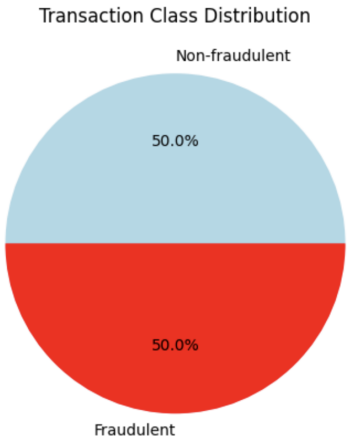

Figure 2. Dataset's class distribution after balancing

Figure 4.2b

This technique has some advantages that could be beneficial. Firstly, it avoids the model's bias as both classes are balanced. Also, it does not contain duplicate instances of minority class, which could cause a problem because it would not introduce any new information and the model cannot learn new features or patterns. Moreover, with this balance it reduces the training time and the memory usage as it has a lot fewer instances of the majority class. However, this technique has some disadvantages. Knowing that it excludes a lot of samples, it can lose information that might be valuable from which the model could learn more (this might be relative as it might take all the valuable information through the selected instances). Furthermore, the model's effectiveness depends on some specific characteristics that may be excluded using this technique during training.

4.3 Feature selection

As the dataset contains 24 features, some of the features can be irrelevant and less informative. Feature selection technique can help as it includes the removal of irrelevant features that can negatively influence the model's performance and can introduce noise to the data. It can benefit the CNN by reducing the risk of overfitting, reducing the dimensionality and reduction of noise.

```
Data columns (total 23 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   Unnamed: 0             1296675 non-null  int64
 1   trans_date_trans_time  1296675 non-null  object
 2   cc_num                 1296675 non-null  int64
 3   merchant               1296675 non-null  object
 4   category               1296675 non-null  object
 5   amt                    1296675 non-null  float64
 6   first                  1296675 non-null  object
 7   last                   1296675 non-null  object
 8   gender                 1296675 non-null  object
 9   street                 1296675 non-null  object
 10  city                   1296675 non-null  object
 11  state                  1296675 non-null  object
 12  zip                    1296675 non-null  int64
 13  lat                    1296675 non-null  float64
 14  long                   1296675 non-null  float64
 15  city_pop               1296675 non-null  int64
 16  job                    1296675 non-null  object
 17  dob                    1296675 non-null  object
 18  trans_num              1296675 non-null  object
 19  unix_time              1296675 non-null  int64
 20  merch_lat              1296675 non-null  float64
 21  merch_long             1296675 non-null  float64
 22  is_fraud               1296675 non-null  int64
dtypes: float64(5), int64(6), object(12)
memory usage: 227.5+ MB
```

Figure 3. Feature datatypes before feature selection

In Figure 3 is shown that there are 12 object data types that are non-numerical data types. This way, when implementing them into the code it threw an error trying to include features like transaction date and time or category that could help the model. That is why we decided to exclude all the object datatypes. So, all the dataset features that will be used in our training dataset and their datatypes are shown in Figure 4.

```
Data columns (total 11 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   Unnamed: 0  1296675 non-null  int64
 1   cc_num      1296675 non-null  int64
 2   amt         1296675 non-null  float64
 3   zip         1296675 non-null  int64
 4   lat         1296675 non-null  float64
 5   long        1296675 non-null  float64
 6   city_pop    1296675 non-null  int64
 7   unix_time   1296675 non-null  int64
 8   merch_lat   1296675 non-null  float64
 9   merch_long  1296675 non-null  float64
 10  is_fraud    1296675 non-null  int64
dtypes: float64(5), int64(6)
memory usage: 108.8 MB
```

Figure 4. Feature datatypes after feature selection

The dataset through these stages:

Dataset	Number of samples	Number of features
Firstly introduced	1296674	24
After feature selection	1296674	11
After dataset balancing	15012	11

4.4 Convolutional Neural Network

Our CNN model architecture has three major layers:

- Convolutional Layer (CL) 1: the first step where the input data goes through feature extraction. The convolutional operations are activated to catch patterns and features. The BN helps to normalise the output of the CL, while the dropout layer is used to prevent overfitting by ignoring 20% of randomly selected neurons.
- Convolutional Layer 2: Continues to refine the features/patterns learnt in the first layer. BN is used again to normalize the output of this CL. Dropout rate increases from 0.2 to 0.8 to prevent overfitting.
- Dense Layers (DL): Includes a flatten layer that converts the 3D output of CL to 1D array. In a fully connected layer/DL every input is connected to each output through a learnable weight [4]. Another dropout layer is used to prevent overfitting with a rate of 0.8. Finally, a dense layer produces a single output.

The CNN architecture is represented in the diagram in Figure 5:

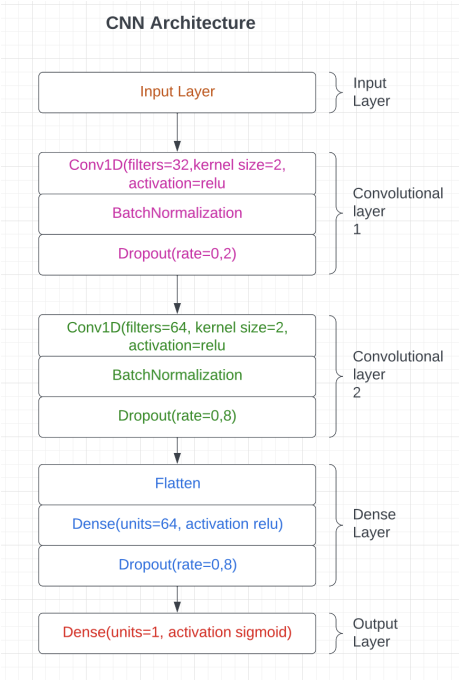

Figure 5. CNN's architecture in this study

4.5 Experiments to improve the original code

The original code used a dropout rate of 0.5 in the second and third dropout layer. Also, it had 20 epochs, did not mention any batch size and had only the accuracy metric.

Learning Curve optimization process

In Figure 6 is shown the first loss/epochs graph that represents the training and validation learning curve (LC) after model's training. The phenomenon of underfitting is appearing.

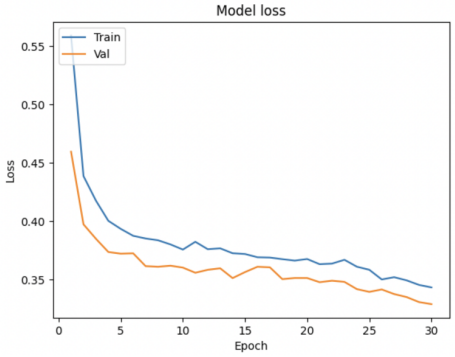

Figure 6. Training and validation loss learning curve after the first training

According to [5], a machine learning algorithm is said to have underfitting when a model is too simple to capture data complexities. It suggests increasing the model's complexity and increase the number of epochs. In the following list are represent the experiments made and the effect they had to model's optimization:

- **Increasing the number of epochs:** The number of iterations of the training process through the entire dataset(epoch) was increased from 20 to 70. This led to an improvement of model's performance during training. The model's downward trend started

to get a horizontal trend, which means the model is improving its predictions.

- **Increasing the dropout rate for the 2nd and 3rd layer:** The original dropout rate was set to 0.5. When decreasing the value, model was overfitting. Same happened when the value was more than 0.85. Finally, after multiple tests decided to increase both rates to the value of 0.8.
- **Changing the learning rate of the optimizer Adam:** The learning rate was originally set at 0.0001. After multiple tests changing the learning rate, realized that the model obtained the best performance using a learning rate of 0.0001.
- **Increasing the batch size:** Batch size was not mentioned. According to [9], if the batch size is not mentioned or is none, it has a default value of 32. After testing different values, result was that an increase in batch size was needed, until a certain level where the underfitting started to increase. The decision was to set the batch size at 192.
- **Adding a pooling layer with pool size 2 to the convolutional layers:** A pooling layer helps in dimension reduction by reducing the spatial dimensions of the feature maps [6]. In the evaluation section, a comparison is done in model's performance and metrics values.
- **Using sigmoid:** For binary classification tasks, sigmoid is better suited. This because it produces an output between 0 and 1 [7]. This is what we want from our CNN model, to produce an output non-fraud (0) or fraud (1).

CNN Model layers orientation hyperparameter values:

Hyperparameter	Value
Epochs	70
Optimizer	Adam
Learning rate	0.0001
Loss	Binary_crossentropy
Metrics	Accuracy, precision, recall, AUC

Figure 7. Table of hyperparameters in our model

5 Results and evaluation

During the testing phase, after the improvements made as explained on section 4.5, the model performed quite well in terms of efficiency. In this section are demonstrated the results obtained from both models (with and without pooling layer), from where a discussion based on the how the model performed and a comparison between both models is included.

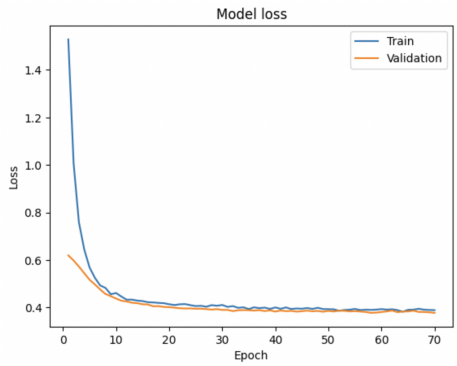

Figure 8. Model's training and validation loss learning curve after training

In figure 8 is represented the training and validation loss of the model 1(without a pooling layer). In first 5 epochs, there is a higher difference between the training and validation LC. After the 5th epoch, both LC follow almost the same trend horizontally, which means the model is not improving the results with huge steps anymore.

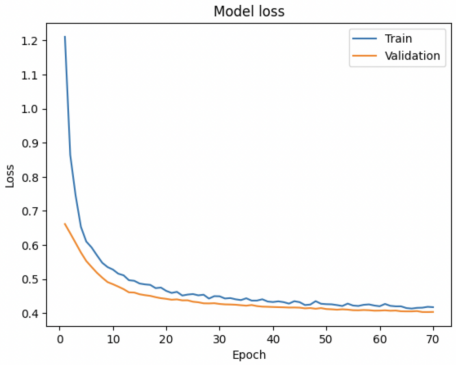

Figure 9. Model using pooling layers's performance

In figure 9 is represented Model 2(includes a pooling layer) training and validation loss. Underfitting is seen through the whole training process. This means that the dimension reduction has not improved the model's performance. Consequently, the proposed model performs better without dimension reduction after each CL.

The results of the confusion matrix for model 1 are represented on in Figure 11. From the matrix we can obtain that:

1. The model correctly predicted 1122 instances as fraud (True Positives in confusion matrix terms),
2. 46 instances were incorrectly flagged as fraud (False Positives),
3. 379 instances were incorrectly predicted as non-fraud (False Negatives),
4. 1456 instances were correctly predicted as non-fraud (True Negatives).

Ultimately, this means that the model makes relatively good predictions.

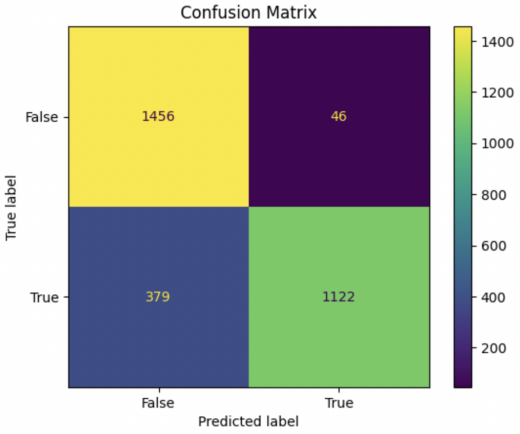

Figure 10. Confusion Matrix

The results of the Table in figure 11 are the metrics that we used to evaluate both models

	Model 1	Model 2
F1-score	0.841	0.819
Accuracy	0.858	0.841
Precision	0.960	0.953
Recall	0.747	0.716
AUC	0.893	0.877

Figure 11.

As mentioned at the learning curve evaluation, the model 1 again has better results than model 2, which means that Model 1 is a better approach, and it works better without a pooling layer in both convolutional layers.
These results are a mirror of the results obtained in the confusion matrix. The accuracy means that the model has correctly categorised as fraud or non-fraud in 85.5% of the instances. Precision measures how many instances were actually fraud out of all instances predicted as fraud by the model, which in our case 96% of them. In terms of recall, the model has correctly predicted 74.7% as fraud out of all fraudulent instances

Furthermore Table in Figure 12 presents the performance comparison of this study with other proposed studies.

REF	TECHNIQUE	ACCURACY	PRECISION	RECALL	F1 SCORE
[4]	CNN+ADASYN	0.09982	0.9976	0.9910	0.9873
[10]	CNN + imbalanced	0.826	Not specified	Not specified	Not specified
[11]	CNN+ Random Oversampling	Not specified	0.822	0.782	0.801
PROPOSED MODEL	CNN+ match	0.858	0.960	0.747	0.841

Figure 12.

From the results in table 5.2, it can be seen that the model proposed in this study outperforms [3] and [5] models, while is being outperformed by [4]s model.

6 Conclusion and future work

A safe and convenient environment is one of the most crucial needs for e-commerce companies. In this study a CNN Model for CCFD is successfully implemented by using a balanced dataset. The performance results show an improvement in the predictive performance after making the process more complex, while there's a decline of performance when using a pooling layer in both convolutional layers. For future study, the performance of the proposed model should be further improved because as seen in [4], there are other studies that have better performance using the same model. Furthermore, exploring other deep learning models can be the focus of future studies.

ACKNOWLEDGEMENTS

I would like to thank my group mate [4] for finding different datasets before choosing the final one. I also would like to thank the tutor Aniket Basu and the lecturers, who helped every week by answering each question or confusion. Also would like to thank all the groupmates to help with different insights or confusion.

REFERENCES

[1] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan, 'Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions', *Journal of big Data*, **8**, 1–74, (2021).
[2] Eyad Abdel Latif Marazqah Btoush, Xujuan Zhou, Raj Gururajan, Ka Ching Chan, Rohan Genrich, and Prema Sankaran, 'A systematic review of literature on credit card cyber fraud detection using machine and deep learning', *PeerJ Computer Science*, **9**, e1278, (2023).
[3] Aya Abd El Naby, Ezz El-Din Hemdan, and Ayman El-Sayed, 'Deep learning approach for credit card fraud detection', in *2021 International Conference on Electronic Engineering (ICEEM)*, pp. 1–5, (2021).
[4] Muhammad Liman Gambo, Anazida Zainal, and Mohamad Nizam Kassim, 'A convolutional neural network model for credit card fraud detection', in *2022 International Conference on Data Science and Its Applications (ICoDSA)*, pp. 198–202, (2022).
[5] GeeksforGeeks. Underfitting and overfitting in machine learning, 2023.
[6] GeeksforGeeks. Cnn - introduction to pooling layer, Year of publication.
[7] krishna rao gaddekrishna rao gaddenbsp. What are the best activation functions for binary text classification in neural networks?, Sep 2018.
[8] Nlgrf. Nlgrf/credit-card-fraud-detection-using-cnn-in-tensorflow-2.0: Credit card fraud detection using cnn in tensorflow 2.0.
[9] Rukshan Pramoditha. Determining the right batch size for a neural network to get better and faster results, Aug 2023.
[10] Imane Sadgali, Nawal Sael, and Faouzia Benabbou, 'Fraud detection in credit card transaction using neural networks', in *Proceedings of the 4th international conference on smart city applications*, pp. 1–4, (2019).
[11] Hilal Nur Tek and Sezen Sude Gul, *kredi karti dolandiricilik tespitinde yapay zeka temelli yontemler*, (2022).
[12] UK Finance. Over £12 billion stolen through fraud in 2022, nearly 80% via app, 2022. Accessed: December 8, 2023.