Credit Card Fraud Detection Using Deep Learning

Olin Business School

Group 12: Alan Li (Leader), Yuki Ao, Cassie Zhang, Christy Ren, Rita Shi

Abstract

Hundreds and thousands of fraud transactions happen every day, leading to not only customers but financial companies millions of dollars. In this report, we will help banks and financial institutions to minimize losses by detecting potential fraud transactions. Our goal is to build a neural network classifier to identify and stop fraud attacks quickly and accurately while improving customer and citizen experiences.

To reach this goal, we apply deep learning neural networks to make predictions on the Credit Card Fraud Detection dataset for detecting the fraud transactions. By building network models, we reached an accuracy of 99.85%. We conclude that by carefully studying the transaction data and taking action before the fraud transactions occur, banks can help card owners prevent over 80% of fraud transactions.

Keywords: Deep learning, fraud detection, supervised learning, machine learning

Introduction

Have you received a call during the day from “Marriott” and got informed that you were selected for a free stay at a 5-star hotel? Congratulations. You are most likely on the way to credit card fraud! In this study, we will use deep learning to help you and your bank stay away from such tricks.

Credit card fraud is very common. It occurs when someone uses your credit cards without your permission, such as for making purchases or withdrawing money. It probably happens when your credit cards are stolen or get lost. Sometimes it can also be stolen “online” - when your personal information gets leaked accidentally when you enter your information online, such as your social security number. Now, credit card fraud can not only happen in person, but also by phone, email, or even social media. According to a survey, more than 270,000 cases of credit card fraud occurred in 2019 alone.

However, nowadays banks and public sectors develop complicated systems for detecting frauds from daily transactions. This will not only help the credit card companies build up their prestige, but also improve the whole society’s trust and efficiency of making transactions online.

The common process of credit card fraud detection and investigation follows the steps below:

1. Monitor the transaction data for each card and watch for suspicious behavior.
2. Use complicated computer algorithms to detect abnormal transactions.
3. Contact the credit card issuer and provide great customer service.
4. Upgrade software and hardware.

In this report, we focused on the second step and designed a deep learning model to detect risky transactions. We first implemented data exploratory analysis to better understand the data. Then we built a baseline network model using Keras with five layers. Then we tried different combinations of hyperparameters and optimizers and got the best model. The hyperparameters include optimizers, activation function, regularization, dropout, and batch size. We use this final model for final conclusion. With this model, we can help the credit card companies and users to lower the fraud transaction risk by 80%. Credit card users and banks will benefit from this technology.

Review of the literature

The fraud detection of credit card transactions has been a topic either in machine learning or the business world. Many machine learning researchers have tried different algorithms and increased prediction accuracy. However, in this specific problem set, there's one problem that cannot be ignored. The transaction data is bound to be highly imbalanced because the fraud transactions are the minority class. However, we need to increase the accuracy of predicting the fraud class.

To solve the problem of learning from imbalanced data, we researched a few related papers and projects. (Zhang, Yazhou, 2018) [1] discussed traditional oversampling algorithms, such as Synthetic Minority Over-sampling Technique (SMOTE) or Adaptive Synthetic Sampling Approach (ADASYN), and their limitations on high dimensional data. The thesis proposed a deep generative model-based imbalanced learning algorithm, where both Variational Autoencoder (VAE) and Generative Adversarial Network (GAN) are implemented as generators for creating data.

The project by (Nash, C., 2017)[4] and (Rajesh Sabari, 2020)[2] used the same approach as above on the same card transaction dataset as ours. GAN was used to generate fraud data, using cross-entropy from the discriminator to train the generator. The GANs with actual and generated fraud data were trained using classic machine learning classifiers and CNN. However, they didn't put much effort into tuning the predictive model.

The study in (Alghofaili, Y et al., 2020)[3] suggested a different approach to the classification models. It proposed a fraud detection model based on the deep learning-based technique Long Short-Term Memory. The author tuned several hyperparameters to improve the results, such as optimizer, batch size, epochs, and matrix. We used the same research logic in our research, too.

Problem description

The two main problems we are facing are:

1. How to overcome the problem of imbalance in an efficient way.

   There is a severe skew in the class distribution. There are 99.833% normal transactions and 0.167% fraud transactions. Many classification algorithms have low predictive accuracy for the infrequent class. Oversampling will duplicate examples in the minority class and thus solve the problem.

2. How to find the best hyperparameters to reach the best prediction result.

   There are so many combinations and it is not impossible to run each model. Therefore, we adopt an asymptotic approach: tune one parameter first to get the optimal value, fix it, and move on to the next one. In this way, we are able to get the optimal value for all hyperparameters.

Data and experimental results

Data Collection

We obtained our dataset from Kaggle (https://www.kaggle.com/mlg-ulb/creditcardfraud). The dataset contains 284,807 credit card transactions that occurred in two days in September 2013 by European cardholders.

The dataset contains 30 numeric predictor variables which are the result of a PCA transformation and 1 target variable indicating whether the transaction is a fraud or not(1 in case of fraud and 0 otherwise). Due to confidentiality issues, the original features and more background information about the data cannot be obtained. Except for features 'Time' and 'Amount', features V1, V2, … V28 are the principal components obtained from PCA. Feature 'Time' represents the time(in seconds) between each transaction and the first transaction in our dataset, which will be transformed into a time of day(in hours) later in our model. Feature 'Amount' is the transaction amount.

Exploratory Data Analysis

Among the transactions, we found 1,062 duplicated normal transactions and 19 duplicated fraud transactions. After dropping the duplicates, we have 473 frauds out of 283,726 transactions. That is to say, the dataset is highly unbalanced since there are 99.833% normal transactions and the positive class (fraud transactions) accounts for only 0.167% of all the transactions.

Since the ‘Time’ feature represents time in seconds, which is hard to interpret, we converted it into the time of day(in hours). After the conversion, we found that normal transactions (non-fraud) usually happen from 8 am to midnight (Figure1). Fraud transactions happen at 2-3 am, noon, and 3-7 pm (Figure1).

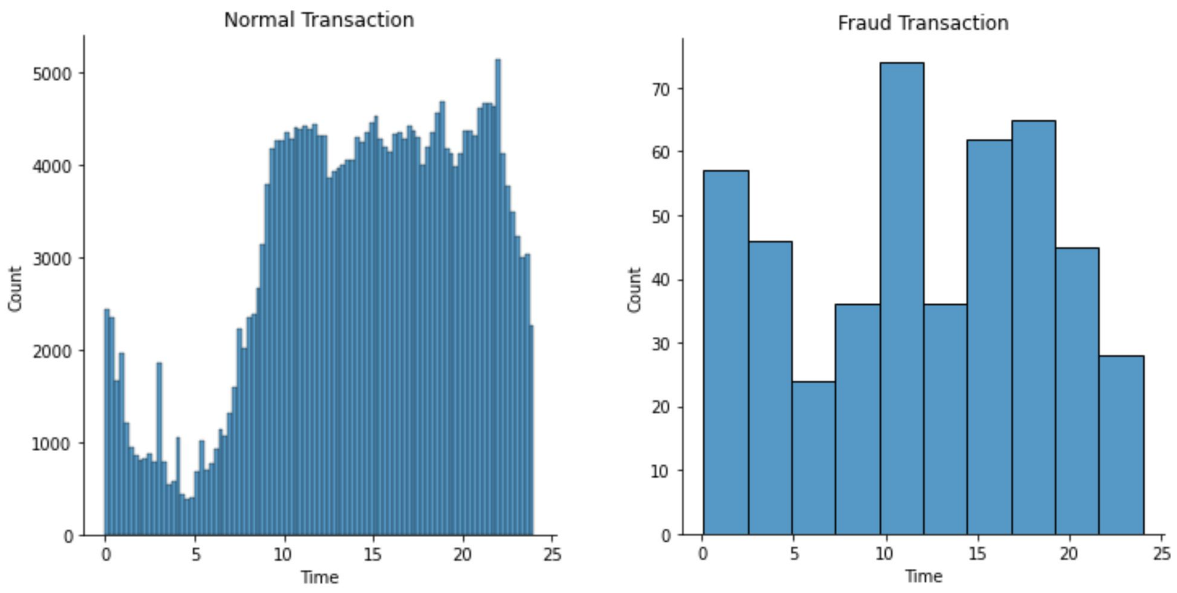


Figure1 Transaction time distribution

We also found that the average value of the transactions is $88.47 and 75% of the transactions are smaller than $77.51. Fraud transactions have an average value of $123.87, 40.1% higher than that of normal transactions($88.41) (Figure 2). However, most of the fraud transactions are small transactions since the majority of them are below $100 and there's even a spike at below $10 (Figure 2).

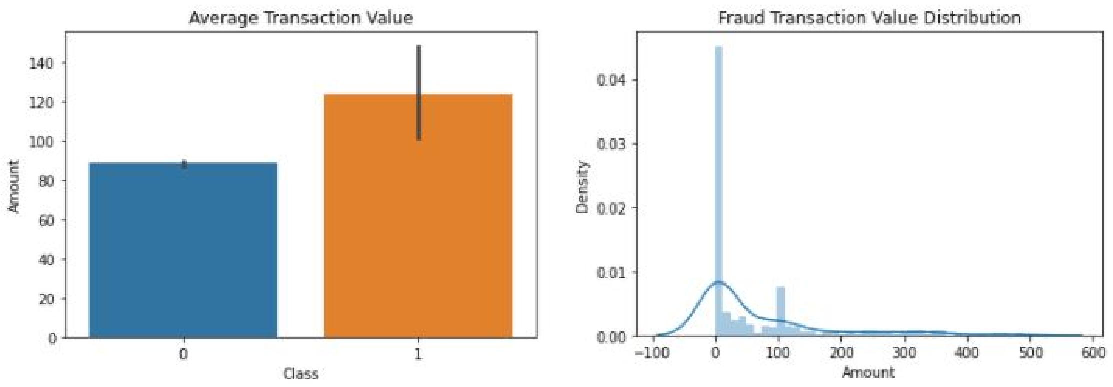


Figure 2 Transaction value

Data Oversampling

The data we are using is not well balanced,after cleaning our data we have 473 frauds out of 283,726 transactions.. The imbalanced data will increase the risk of overfitting of the regular activity and underestimate fraud activities. We realized that there are a few ways to generate more data for fraud activities. There are two methods that we discussed in the process of building the models. Firstly, Generative Adversarial Networks(GAN) is the method that we are willing to try on our model. As we have mentioned in the literature review section, GAN can adapt the original dataset's features and generate new data. As we have mentioned in the first literature that we have looked into, GAN will have higher accuracy with higher dimensional data. However, the computation time and the limitation of our technique have tied our hands. We convert to another method, Synthetic Minority Over-sampling Technique (SMOTE). SMOTE is also generating new data by using the features' nearest neighbors. This computation is much more time-efficient and commonly used. We decided to use this technique in the latter part of our study to supplement the GAN method.

Model description

We started with a simple neural network with **4 layers**, each of which contains 128, 64, 32, and 1 filter. After a few tryouts, we found that the model is underfitting, with validation loss always smaller than training loss. Therefore, we increased the number of layers to **5** (which contains 512, 128, 64, 32, and 1 filter) and **6**, when we found models with 5 layers have neither overfitting nor underfitting problems but have the best F1 scores and accuracies.

Before we built the model, we used **SMOTE** to oversample the data. We also compared the oversampled model with the one without oversampling. It turned out that the model without oversampling performs slightly better.

Then, we compared different **optimizers** of Adam, RMSprop, and Adagrad with the baseline model (the model with 4 layers). Based on the chart listed below, we can clearly see that Adam optimizer yields the best F1 score and the best accuracies. Therefore, we use Adam for all of our later models.

Result of experiment with different optimizer									
Model Number	Optimizer	Activation Function	Regularization	Dropout	Batch Size	Number of layers	Fraud F1-score	Training accuracy	Validation accuracy
1	Adam	Relu	/	/	100	4	0.68	0.9987	0.9982
2	RMSprop	Relu	/	/	100	4	0.53	0.9976	0.9971
3	Adagrad	Relu	/	/	100	4	0.29	0.9884	0.9929

Chart 1 Result of experiment with different optimizer

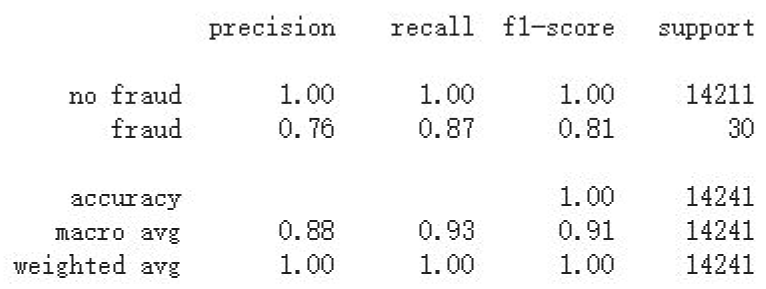
We used **early stopping** monitors for all the models as well.

For every model, we tried different combinations of hyperparameters:

- **Activation function**: Relu and sigmoid. We soon found that Relu gives better results.
- **Regularization**: we applied L2 regularization, which poses negative influences on the results.
- **Dropout**: setting a dropout at an arbitrary level (e.g. 0.1) will improve the F1 score. Even for the model that is underfitting, dropout will increase the F1 score as well.

- **Batch_size**: we tried batch_size from very small numbers (e.g. 16) to larger numbers (e.g. 512). It turned out the best batch_size should be between 100 and around 300.

For every model, we generate a classification report as demonstrated below:



```
              precision    recall  f1-score   support

    no fraud       1.00      1.00      1.00     14211
       fraud       0.76      0.87      0.81        30

    accuracy                           1.00     14241
   macro avg       0.88      0.93      0.91     14241
weighted avg       1.00      1.00      1.00     14241
```

Figure 3 Prediction Result

Here, the F1 - Score for “fraud” is what we focus on, because we care more about fraud detection. Precision and recall are very close to 1 but they are actually not 1.

(chart of the excel Linked to google excel, will be semi-automatically updated (Alan))[1]

Result of experiment with different hyperparameter

Model Number	Optimizer	Activation Function	Regularization	Dropout	Batch Size	Number of layers	Fraud F1-score	Training accuracy	Validation accuracy
1	Adam	Relu	/	/	100	4	0.68	0.9987	0.9982
2	Adam	Sigmoid	/	/	100	4	0.48	0.9975	0.9956
3	Adam	Relu	/	/	50	4	0.5	0.9986	0.998
4	Adam	Relu	/	/	10	4	0.52	0.9978	0.9978
6	Adam	Relu	/	/	150	4	0.53	0.9987	0.998
5	Adam	Relu	with reg	/	100	4	0.64	0.9992	0.9991
7	Adam	Relu	/	0.1	100	4	0.77	0.9979	0.9988
8	Adam	Relu	/	0.05	100	4	0.74	0.9987	0.9988
9	Adam	Relu	/	0	100	5	0.81	0.9993	0.9985

[1] (Only insightful models are shown here.Red fonts indicate change of variables, while yellow backgrounds indicate best results.)

10	Adam	Relu	/	0	100	5	0.74	0.9989	0.999
11	Adam	Relu	/	0	16	5	0.58	0.9975	0.9972
12	Adam	Relu	/	0	32	5	0.5	0.9979	0.9968
13	Adam	Relu	/	0	64	5	0.69	0.9987	0.9987
14	Adam	Relu	/	0	128	5	0.69	0.9991	0.998
15	Adam	Relu	/	0	256	5	0.75	0.999	0.9987
16	Adam	Relu	/	0	512	5	0.66	0.9991	0.9986
17	Adam	Relu	/	0	100	5	0.76	0.9995	0.9997
18	Adam	Relu	/	0.1	128	5	0.68	0.9989	0.9984
19	Adam	Relu	/	0.1&0.05	128	5	0.78	0.9989	0.9992
20	Adam	relu	/	0	100	6	0.74	0.9989	0.9984

Chart 2 Result of experiment with different hyperparameter

We chose the model because our data is simply made up of numbers. So we would not need to apply CNN here. GANs would be great for oversampling, but it is extremely time-consuming. We would not have enough time to apply GANs. Plus, the results without oversampling are better. (We would still need to use oversampling in the end, on account of making the most sense.)

If we have more time, we would definitely try GANs to oversample. We would also try more selections of hyperparameter values, especially dropout ratios at different layers. The simple choice of 0.1 already makes the model better, and with more time we should be able to find a certain range of dropout ratios that brings the best results.

Conclusions, Discussions, and Recommendations

The best model we generated reached a validation accuracy of 99.85%. The model that we create can help credit card companies supervise transactions and have 99.85% of the chance to identify fraud transactions. The accuracy for detecting true fraud activity is 81% (F1- score), which means with the help of our model credit card companies can lower the risk of fraud transactions by 81%.

The oversampling for Fraud will make the F1 score drop. This may be caused by the severe imbalance of the data - There are too many normal activities so if we are predicting for normal transactions, we will still get a good prediction result. However, it is still important to do oversampling to get a better dataset with less noise ratio.

There are some models that give better accuracy rates, but F1-scores of these models will be dropping at the same time. The fact the accuracy rates do not outperform significantly better is sufficient for us to identify the most crucial changes and findings we observed, which are listed below:

1. We compared the **number of layers** in the beginning and found out that the model with 5 layers is the best. It either has an overfitting problem (case for the model with 6 layers) or an underfitting problem (case for the model with 4 layers), represented by validation loss staying above/below training losses.
2. **Adam** appears to be the best optimizer, especially for a general all-number question like this. This is also validated by many academic works.
3. **Regularizer** tends to bring a negative influence to the metrics.
4. Adding a **dropout** at any arbitrary level (e.g. 0.1) will improve the F1 score. And this effect seems to be uninfluenced by the underfitting problem. We would normally use dropout when there is an overfitting issue. However, even for the model that is underfitting (model with 4 layers), dropout will increase the F1 score as well.
5. F1-score (and accuracy rates) increases with **batch_size** (from 16 to 256) and starts to decrease after batch_size reaches 256. It appears that optimal batch_size is within a range of (100, ~300). Because when batch_size = 512, all the metrics get worse. A batch_size value, in this case, of more than 300 should be considered relatively too large. Therefore, one of the future topics is to tune the best batch_size.

Here are some aspects we can improve with more time:

1. The dataset is too small and imbalanced, so the normal transactions are very likely to be detected. However, the fraud ones are very hard to detect. If we can get larger data, the result may be more accurate. Our original data only contains transactions in two days in September 2013. The best option is recollecting data by using the data from 2013 till now. This will increase our data and enlarge the fraud pool.
2. The method of oversampling that we used is SMOTE, which may not be the most accurate way of generating fraud transactions in the training dataset. A more accurate oversampling method is required, e.g., using GANs to expand fraud data.
3. The model is too simple. From the chart below, the yellow line is the validation accuracy. It can be seen that the yellow line can be higher than the blue, which means that the model could be better in prediction accuracy. We need more complicated models.
4. Look for the best dropout ratio and place (which layer the dropout should be added to).
5. Tune the best batch_size.

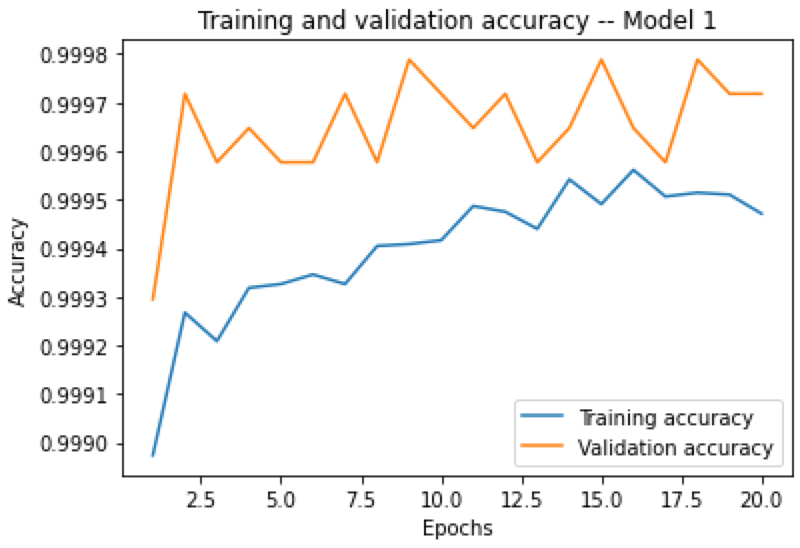


Figure 4 Model 1 Accuracy Curve

Fraud detection certainly is an important business problem. And we started with high validation accuracy values already. However, we were not satisfied with that and we kept looking for a deeper understanding of the problem, and we identified key variables that would shift the key metric F1-score. Based on these findings, we also proposed future moves.

References:

1. Zhang, Yazhou. (2018). *DEEP GENERATIVE MODEL FOR MULTI-CLASS IMBALANCED LEARNING.* Open Access Master's Theses. Paper 1277. https://digitalcommons.uri.edu/theses/1277
2. Sabari, Rajesh. (2020). *Improving Credit Card Fraud Detection based on CNN/GAN.* http://cs230.stanford.edu/projects_winter_2020/reports/32635168.pdf
3. Alghofaili, Y., Albattah, A., & Rassam, M. A. (2020). *A Financial Fraud Detection Model Based on LSTM Deep Learning Technique*. Journal of Applied Security Research, 15(4), 498-516.
4. Nash, C. (2017, November 14). *Create Data from Random Noise with Generative Adversarial Networks*. Toptal Engineering Blog. https://www.toptal.com/machine-learning/generative-adversarial-networks

Team contribution:

1. Data collection: All;
2. Literature review: Yuki;
3. Data preprocessing and EDA: Christy, Yuki, Cassie;
4. Model building and improvement: All;
5. Results and insights: All

6. Report: All.