Kaggle IEEE-CIS Fraud Detection Challenge

Chintana Prabhu - cprabhu@sfu.ca
Jason Lee - jason_lee_19@sfu.ca
Wesley Romey - wes_romey@sfu.ca

Abstract

More transactions are becoming electronic and original fraud prevention methods are no longer relevant, thus machines are being used to prevent fraud. Convolutional neural networks (CNN) have proven to be effective in not only image related work but also in fraud detection for transactions. Detecting fraud can be accomplished using machine learning. The main metric focused on in this report is the area under the ROC curve. A machine learning algorithm composed of feature extraction, data preprocessing / cleaning of a Kaggle [1] dataset, and a CNN is implemented. This CNN has achieved an AUC of 0.873. These results are compared to other works that use machine learning to detect fraudulent transactions. While not quantitatively and directly comparable, it follows that techniques such as implementing a neural network ensemble [2], [3] and feature shuffling [4] would be beneficial for future projects.

1 Introduction

Swindlers have been around for years and transaction fraud has been an unignorable issue. Up until 2015, fraudsters would be able to simply copy numbers from a card to complete transactions but recently cards have started using chips allowing retailers to demand PIN. Additionally, merchants have started truncating account numbers on receipts, checking IP addresses, and tokenizing card information before storing [5].

While these implementations have taken place, swindlers have thought of various other ways to steal identity and will continue finding new methods. As a result, credit card fraud has become the most common issue in identity theft [6]. With $27.85 billion in losses due to transaction fraud recorded in 2018 [7], fraudulent losses are becoming more prevalent than ever before and are projected to continue increasing.

To combat this issue, machine learning can be used since a lot of fraudulent transactions have the same or similar characteristics and a program can quickly pick up on those patterns. Hence, it is important to find a good machine learning algorithm to detect fraud reliably and convolutional neural networks in particular are very effective for this problem. The following sections of this report include the objectives, related works, methodology, results, discussion, and the conclusion.

2 Objectives

The main objective is to detect as many of the fraudulent transactions as possible using machine learning. The main indicator of accuracy we use to measure neural network performance is the area under the ROC curve (referred to as AUC in this report).

3 Related Work

Zhang et al. [4] incorporates feature sequencing into their algorithm for real-time fraud detection. Feature sequencing involves training multiple neural networks, each with a different ordering of the features [4]. Since testing for all possible feature combinations for a large number of features is impossible, the ordering is randomly selected and a fixed number of neural networks are trained [4]. The best-performing neural network is chosen as the final network [4]. The results are measured in terms of the precision and recall. These are measured for 6 sets of data and both the precision and recall are around 90% for all datasets [4], whereas the accuracy hovers at around 99% [4]. Zhang et al. [4] uses a neural network design where the feature sequencing is done prior to training each neural network. Then, a 1D convolution layer followed by a 1D pooling layer followed by a fully connected (dense) layer are used. The final layer is the output layer. This is similar to the neural network layer structure presented in the methodology section, except less complicated.

Another possible technique which can be combined with feature sequencing is given by Carta et al. [3]. Carta et al. [3] employs a neural network ensemble method for fraud detection, where multiple relatively simple networks are combined using statistical techniques to form an improved prediction. According to Granitto et al. [2], for neural network ensembles to deliver a benefit, it is important for these neural networks to be diverse in their structure, data, and / or training method. In [3], a Prudential Multiple Consensus model is used, where the diversity of the ensemble comes from varying the algorithms used to train the data [3]. The state-of-the-art classification algorithms are used. Combining the adaptive boosting (ADA), gradient boosting (GBC), gaussian naive bayes (GNB), multilayer perceptron (MLP), and random forests (RFC) algorithms in an ensemble resulted in an AUC of 0.884 [3]. However, combining only GNB and RFC improved the AUC to 0.898 [3]. However, according to [3], using each algorithm alone gave the following AUC values:

- ADA: 0.651
- GBC: 0.808
- GNB: 0.862
- MLP: 0.576
- RFC: 0.896

In Carta et al. [3], the improvement was marginal because RFC is significantly better than GNB in [3] and both are significantly better than any of the other tested algorithms [3]. Hence, it is practical to leave relatively bad neural networks out of the ensemble. It is worth noting that [3] used less data than this report uses, as [3] uses approximately 300 thousand datapoints with approximately 500 being fraudulent. This leaves room for overfitting. However, the database from Kaggle [1] contains approximately 500 thousand transactions with approximately 20 thousand being fraudulent transactions. The method of combining each of the individual models used by Carta et al. [3] is called "majority voting", where a threshold is chosen between 0 and 1 and the average of the outputs by all algorithms in the ensemble is compared to the threshold. [3]. If the threshold is exceeded, then the transaction is considered fraudulent, otherwise, it is not considered fraudulent.

4 Methodology

A machine learning algorithm is used to detect fraud. This algorithm can be accessed at [https://github.com/chintanaprabhu/CMPT726-CJW]. This code can be run using any dataset with few edits required, such as edits to the file names and to various constants. At a high level, the code uses data from Kaggle [1], though this data can be replaced with any identically-formatted data of any size with minimal edits to the code. The data is cleaned and preprocessed. Next, the features are extracted, where some new features are created and other features are removed from the original Kaggle dataset [1]. A neural network model is then created and trained. Finally, the model accuracy,

loss, and AUC are plotted. Additionally, an additional test dataset from Kaggle [1] is processed to generate predictions about fraudulent transactions.

4.1 Feature Extraction

Relative to the original Kaggle dataset [1], several new features are created and other features are removed. These new features include the day of the transaction relative to an arbitrary reference time and the hour of day the transaction occurs. The logarithm of the amount of money transferred in the transaction is used instead of the actual transaction amounts to prevent the extremely large values from dominating. 54 features were removed from the dataset before training began. Any feature where at least 90% of the values had the value of NaN was removed. Also, those where one value repeats itself more than 90% of the times were also removed. The slight trend of fraud occurring in the morning much more than in the evening is apparent in this dataset, as shown below:

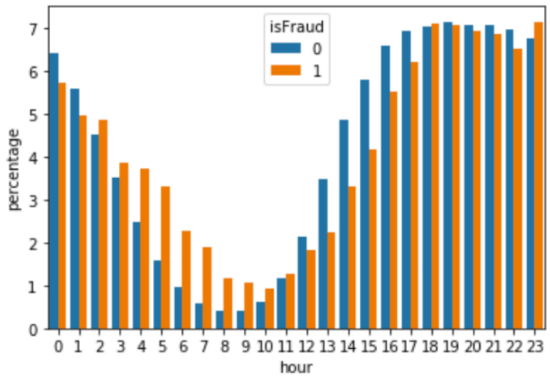

Figure 1. Transaction fraud as a function of hour.

An illustration of the transaction amounts averaged over all data points can be seen in the following figure:

	TransactionAmt
count	590540.000000
mean	135.027176
std	239.162522
min	0.251000
25%	43.321000
50%	68.769000
75%	125.000000
max	31937.391000

	TransactionAmt
count	506691.000000
mean	134.725568
std	245.779822
min	0.018000
25%	40.000000
50%	67.950000
75%	125.000000
max	10270.000000

Figure 2. TransactionAmt Feature, describing the transaction amounts for the training data [1] (left) and a separate validation dataset from Kaggle [1] (right).

4.2 Data Cleaning and Preprocessing

The dataset consists of two tables (transaction and identity) that are merged into one dataframe. Then, the data is looped to check for and remove features containing mostly NaN values. Features are then normalized and the remaining NaN values are replaced by mean or mode depending on if they are numerical or categorical respectively. Finally, features are sorted to complete data cleaning (Figure 3).

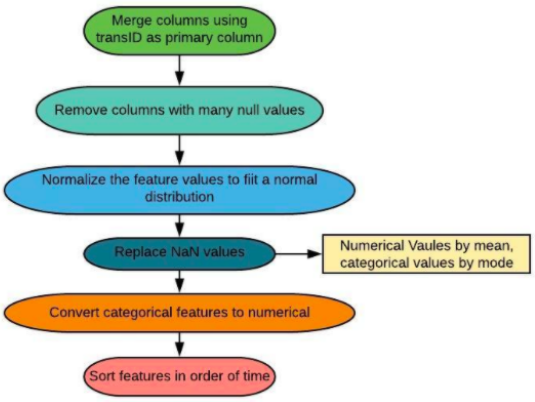

Figure 3: Data Cleaning Steps

The data is separated temporally into train and validation sets, where transactions that occur before a certain point in time are in the train set and values occurring later are in the validation set. The data is oversampled so the number of fraudulent and non-fraudulent transactions are equal in the database.

4.3 Convolutional Neural Network (CNN)

The CNN involves several layers. The main layers are two 1D convolution layers with 20 nodes and a kernel size of 2 and 2 dense layers. The neural network also involves a flatten layer, batch normalization, max pooling, and regularization in the form of dropout layers. The summary of how each layer works is summarized in the following table:

Table 1: Description of neural network layer types used in this report.

Layer Type	Description
1D Convolution	A layer where all nodes are connected to some of the nodes from the previous layer [8]. This is defined by a kernel size, which is the number of nodes from the previous layer each node is connected to [8].
Dense	A layer where all nodes in the layer are connected to all nodes of the previous layer. An activation function is then applied to each layer node element-wise, which creates the output of each layer [8].
Batch Normalization	Normalizes the outputs of the previous layer for each batch such that the average output is near 0 with a variance of 1 [8].
1D Max Pooling	Reduces the dimensionality by looking at a group of nodes from the previous layer and saving the maximum value to the output [8].
Dropout	Regularization technique where a fraction of nodes in a layer are dropped out, encouraging every node in a layer to be used [8].
Flatten	Changes dimensionality of the layer to facilitate a transition from one type of neural network layer to another [8]. In this report, the transition is to a dense layer. Without the flatten layer, the code for this report does not work properly.

In total, there are 38681 trainable parameters in the network which are mostly due to the connection between the flatten and dense layers. The activation functions are all RELU except for the last layer, where the activation function is sigmoid. This is because the last layer has a value between 0 and 1, where 0 indicates no fraud and 1 indicates fraud.. The following figure summarizes the neural network at a high level:

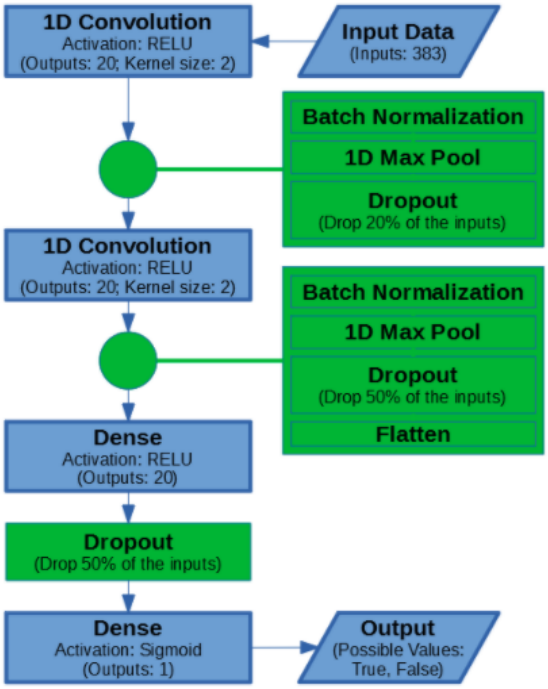

Figure 4: CNN Design

4.4 Dataset

The dataset was provided by IEEE Computational Intelligence Society from Vesta's real world e-commerce transactions. It consists of two tables, transaction and identity, joined by an ID column. The dataset consists of basic information such as transaction amount, payment card info, and address, but it also contains more advanced features such as counting columns which hold the addresses related to the payment card [9] and match columns which contain matching names and addresses [9]. The dataset is made up of 3.5% fraudulent, and 96.5% non-fraudulent transactions; additionally, train and test transactions timelines don't overlap. Overall, there are approximately 400 input categories in this dataset, most of which are kept confidential for privacy reasons. We used 2 datasets from Kaggle [9]. One of these is approximately 500 thousand data points used to train and validate the neural network as it is being trained. The other is around 500 thousand data points without the information about whether the transactions are fraudulent. Hence, this dataset was used to predict the true output and then was submitted to Kaggle [9], which evaluated the AUC.

5 Results

As measured by the main data from Kaggle [1], the result is the ROC curve has an AUC of 0.873. The train and validation accuracy curves are also shown by epoch, in addition to the train and validation loss. Using a separate test dataset from Kaggle [1] returns a lower AUC of 0.741 [1]. The loss function is evaluated by The following figure summarizes the results with 6 epochs of training:

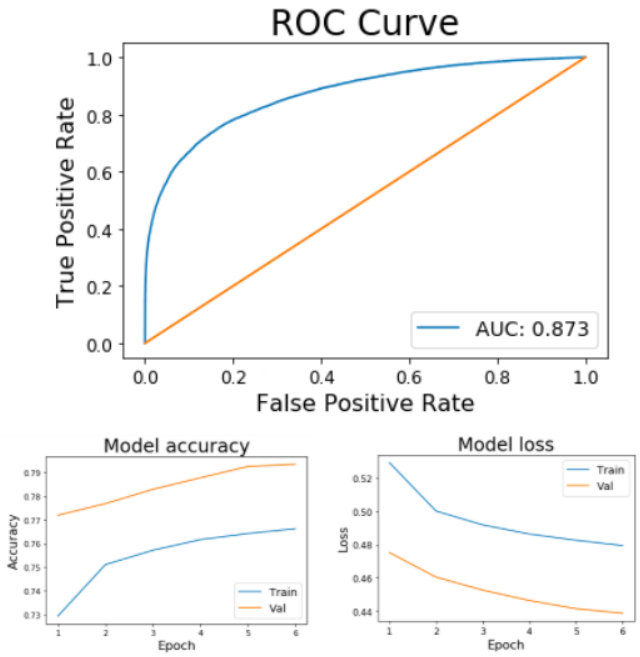

Figure 5: (Top) ROC curve, where the diagonal line is where TPR = FPR and the blue line is the actual ROC curve. (Left) Accuracy by epoch. (Right) Loss by epoch.

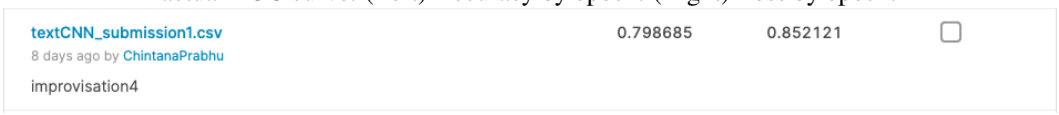

Figure 6: Accuracy value provided by Kaggle on test data

5.1 ROC Curve Definition

The true positive rate (TPR) is the fraction of detected fraud transactions that are correctly identified as fraud. The false positive rate (FPR) is the fraction of detected fraud transactions that are not actually fraudulent. The receiver operating characteristic (ROC) curve is a plot that compares the TPR with FPR. The area under the ROC curve (AUC) is a good way to measure neural network performance, where a larger AUC denotes a better algorithm.

6 Discussion

CNNs are effective for detecting fraud as transaction patterns are correlated to edges. Additionally, running numerous algorithms that improve the accuracy is not necessarily better since some algorithms make little to no difference while taking up time and resources. Regularization is very important for minimizing variance as well as minimizing overfitting and underfitting. Prediction accuracy can be skewed due to biased data and may not be the best metric to use when analyzing the models ability to predict, in some cases it even hides the models inability. The neural network is trained with the explicit purpose of minimizing binary cross entropy, where the model accuracy is a rough indicator of progress. The loss would have kept decreasing (and the accuracy / ROC would have kept increasing) if the neural network was trained for more epochs, but we did not have enough time to run the extra epochs.

There are many metrics by which to measure the quality of a fraud detection algorithm, including accuracy, AUC (for the ROC curve), precision, and recall. Accuracy is commonly used, but does a poor job at giving a fair evaluation for fraud detection algorithms because of the highly unbalanced nature of the raw data. Precision is the probability that the transaction is fraudulent if the algorithm

detects fraud and recall is the fraction of fraudulent transactions that are detected. ROC is the most important one.

As mentioned in the introduction, the strategies used in Carta et al. [3] and Zhang et al. [4] work well. Zhang et al. [4] uses a feature sequencing strategy and a simpler neural network. Though our results cannot directly be compared to them, it is fair to say that, when implemented correctly, feature sequencing has a noticeable benefit, though it is not game-changing by itself [4]. Carta et al. [3] uses a neural network ensemble strategy with the method of diversity being the use of a different training algorithm for each neural network. This strategy theoretically works well, but was barely any better than the best single algorithm tested, random forests (RFC) [3]. This is largely due to the fact that this algorithm was better than the others by far, limiting the benefits of combining multiple algorithms. Since Carta et al. [3] uses a different dataset than us, it is not possible to make direct comparisons, but it is safe to say that neural network ensembles have a benefit when many similar-quality neural networks combine their inputs to form a more accurate conclusion. Granitto et al. [2] indicates that any diverse set of neural networks can produce the benefits of a neural network ensemble if implemented optimally. If the feature sequencing from Zhang et al. [4] and the neural network ensemble algorithms indicated in Carta et al. [3] and Granitto et al. [2] are implemented using our dataset, we would almost certainly have better results.

7 Conclusion

From our experiments with different models and techniques proposed to tackle this problem, we found that using CNNs to classify the transaction as fraudulent or genuine yields good results with adaptation of minimal feature extraction. In order to further improve the model, we can try to implement a neural network ensemble by training many neural networks and combining them in a statistical way to predict the final output. We also believe that running the model for more time on a powerful machine would have improved the performance of prediction. Tradeoff has to be made between time and accuracy. A neural network ensemble would have provided us with better results at the same time CNN consumed less time to provide significant performance.

Acknowledgments

The main inspiration for this project comes from Kaggle [1]. Additionally, Google Collab was used to run the code some of the time. TensorFlow, Keras, Matplotlib, and Scikit-Learn are libraries used in the code. The other notable imports used in the code are csv, sys, os, pandas, numpy, seaborn, gc, imblearn, random, google.colab, and sklearn.

Group Member Contributions

Task	Chintana Prabhu	Jason Lee	Wesley Romey
Research	25%	25%	50%
EDA (software)	100%	0%	0%
Data preprocessing (software)	100%	0%	0%
CNN (software)	60%	0%	40%
Software (other incl. other drafts)	40%	0%	60%
Proposal	40%	30%	30%
Poster	30%	40%	30%
Report	20%	40%	40%

References

[1] Lynn@Vesta, "IEEE-CIS Fraud Detection," Kaggle, 3 October 2019. [Online]. Available: https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203. [Accessed 20 April 2020].

[2] P. M. Granitto, P. F. Verdes, and H. A. Ceccatto, "Neural network ensembles: evaluation of aggregation algorithms," Artif. Intell., vol. 163, no. 2, pp. 139–162, Apr. 2005, doi: 10.1016/j.artint.2004.09.006.

[3] S. Carta, G. Fenu, D. Reforgiato Recupero, and R. Saia, detection for E-commerce transactions by employing a prudential Multiple Consensus model," Journal of Information Security and Applications, vol. 46, pp. 1322, Jun. 2019, doi: 10.1016/j.jisa.2019.02.007.

[4] Z. Zhang, X. Zhou, X. Zhang, L. Wang and P. Wang, "A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection," Hindawi, 2018.

[5] Wikipedia, "Credit card fraud," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Creditcardfraud. [Accessed 20 Apr 2020].

[6] L. Daly, "Identity Theft and Credit Card Fraud Statistics for 2020," The Ascent, 16 Apr 2020. [Online]. Available: https://www.fool.com/the-ascent/research/identity-theft-credit-card-fraud-statistics/. [Accessed 20 Apr 2020].

[7] The Nilson Report, "Payment Card Fraud Losses Reach $27.85 Billion," CISION PR Newswire, 29 Nov 2019. [Online]. Available: https://www.prnewswire.com/news-releases/payment-card-fraud-losses-reach-27-85-billion-300963232.html. [Accessed 17 Apr 2020].

[8] François Chollet, Google, Microsoft, et al., "Keras: The Python Deep Learning library," GitHub, [Online]. Available: https://keras.io/. [Accessed 20 Apr 2020].

[9] IEEE-CIS Fraud Detection Society, "IEEE-CIS Fraud Detection," Kaggle, 23 Jul 2019. [Online]. Available: https://www.kaggle.com/c/ieee-fraud-detection. [Accessed 20 Apr 2020].

[10] S. Narkhede, "Understanding AUC - ROC Curve: What is AUC-ROC Curve?," Medium, 26 Jun 2018. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. [Accessed 17 Apr 2020].