# Improving Credit Card Fraud Detection based on CNN/GAN

Rajesh Sabari; rajeshsa@stanford.edu

## 1. Introduction

Number of frauds in card transactions specifically in online world has dramatically increased recently and traditional Machine Learning algorithms based on supervised binary classification systems are widely prevalent (such as Random forest and GBoost). Such traditional ML algorithms may have a good score in confusion matrix but have poor F1 score. However, in such a specific application domain, datasets available for training are strongly imbalanced, with the number of fradulent transactions considerably less represented than the other. This significantly reduces the effectiveness of binary classifiers, undesirably biasing the results towards the prevailing class, while we are interested in the minority class. Oversampling the minority class is one approach to mitigate this problem but it still has its drawbacks. Having a high precision-recall metrics is a key objective of this project and employing a powerful Deep learning approach (through Multi layer perceptron, CNN) should fare better. I have also trained a GAN to output mimicked fraudulent examples, which were then merged with training data into an augmented training set so that the overall effectiveness of a classifier can be improved.

In this project, I have trained each GAN for 5000 rounds and examined the results along the way (to generate fraud data). My first GAN pits the generator network against the discriminator network, making use of the cross-entropy loss from the discriminator to train the networks. This way is able to learn the shape and range of original data. I have also passed this augmented image through 1x29 convolutional layer (kernel size 29) followed by a fully connected dense layers (300/100 neurons each) to finally have a Softmax prediction $\hat{y}$ determining fraudulent or not.

## 2. Related Work

### Card Prediction using Machine Learning

There have been many attempts in predicting card fraud using various Machine Learning Algorithms. For example, Detecting fraudulent credit card transactions with supervised learning (2018) is able to accurately identify fraudulent transactions using a random forest model

### Analyze Types of Credit Card Frauds

For example, Credit Card Frauds in Banking (2014)
explores the credit card fraud and methods of it, and gives information about what to do in case of encountering credit card fraud by chargeback topic.
In this paper it is studied on the types of credit card fraud such as, application fraud, lost sto len cards, account takeover, fake and counterfeit cards. Also it includes parts of gaining infor mation by taking reports and data from different and safe official sources. Besides that, pap er investigated about how often the occurrence of these methods.

Although these papers and many more employ traditional Machine learning for fraud prediction and prevention – they do not address class imbalance and sparsity of data. The Deep learning algorithms I have trained in this project has a superior performance.

## 3. Datasets

https://www.kaggle.com/mlg-ulb/creditcardfraud

The datasets I have used for training in this project are from Kaggle. The dataset from Kaggle has 31 features, 28 of which have been anonymised are labeled V1 through V28. The remaining three features are the time and amount of transaction as well as a label whether that transaction was fraudulent or not. The variables have been anonymised (as these are actual European card holder transactions) in the form of a PCA. The dataset contains 284,807 transactions. The mean value of transactions is 88.35 USD and the largest transaction value is 25,691.16 USD. Most of the transactions are quite small as you would expect in everyday transactions.The time is recorded in number of seconds elapsed since the first transaction in the dataset. The transactions are over a period of 2 days. 99.83% of transactions in this dataset were not fradulent while only 0.17% were fradulent. There is also minimal correlation between variables – This may be as a result of PCA transformed variables. Hence I dont need to account for any multi collinearity in my model

**Data Pre-Processing & Augmentation**

I have decided to drop time in my initial analysis as I dont have much info on the exact time of transaction. I could speculate most of the fradulent is at night but I have decided to drop the variable in the initial analysis. I have also normalised the amount variable (Standard Deviation) due to large variance. Train-Dev-Test split distribution in my project is 60-20-20 as the number of observations isnt necessarily large. I have also used a 5 split stratified sampling sss1 = StratifiedShuffleSplit(n_splits=5, test_size=0.2, random_state=42). I also performed oversampling due to very few fraudulent data available ada = ADASYN(sampling_strategy='minority', random_state=42). Negative and positive correlations were observed in the balanced dataset. Eg: Features, V14, V12, V10 and V3 show negative correlation towards the 'Class',The balanced training data (oversampled) has 700 batches and 40 epochs.

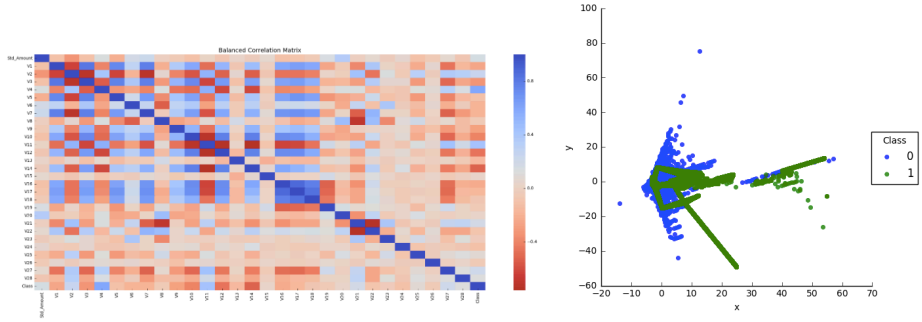



Figure 1: Balanced Correlation Matrix and Balanced Dataset visual representation

Trained GAN was also able to identify fradulent transactions pattern/range between V17, V10 variables and mimic it.

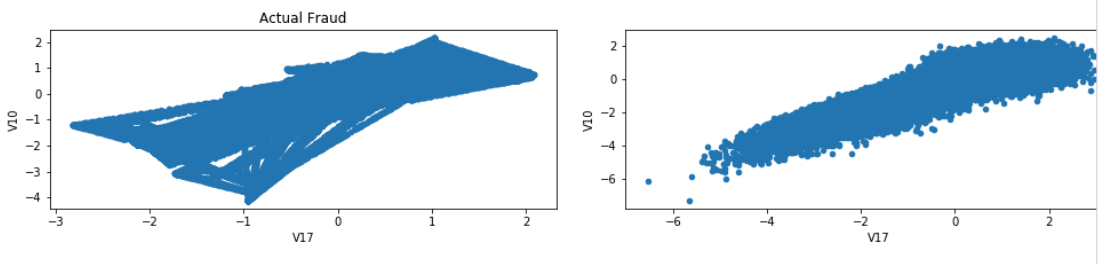
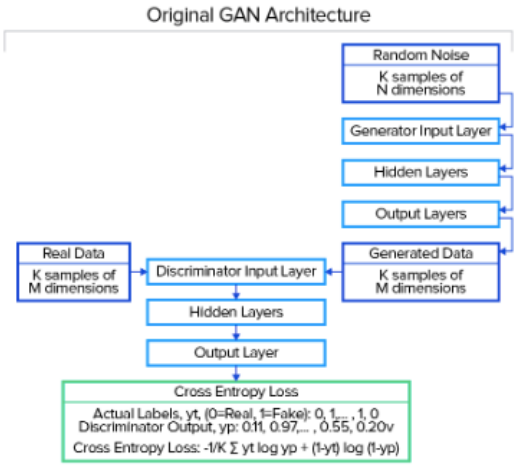



Figure 2: Actual Fraud observations versus Generated (V10 versus V17 variable)

## 4. Methods and Models

The base line models were Logistic Regression, Random Forest and GaussianNB which have already been built and were tested. I initially employed a multi layer perceptron (both 1 and 2 hidden layers) for 100 randomly selected iterations. The first dense layer had 65 neurons. Drop out was also employed to minimise train-val variance. F-score was used as criteria to evaluate performance. I also have trained GAN to enhance the fraduluent dataset by minimising cross-entropy loss (defined as a measuer of how accurately the discriminator identified real and generated images). The augmented dataset was passed through first conv layer (32 filters with width 5) followed by a fully connected dense layer (64 neurons) and finally through a softmax prediction. The credit card dataset lacks any spatial structure among the variables, so I've converted the convolutional networks to networks with densely connected layers

The three deep learning models I have used in this project are 2 layer MLP, GAN and two 1D Conv layer (kernel size 29) , max pooling and a fully connected dense layer (tested with 65 and 300 neurons) terminating with a softmax function (2 class prediction). 2000 epochs were run with ADAM optimization and the loss function for GAN as defined below.

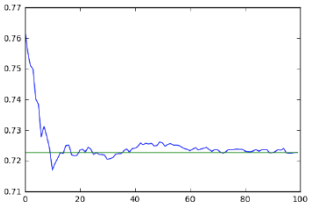



| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_25 (Dense) | (None, 256) | 25856 |
| leaky_re_lu_18 (LeakyReLU) | (None, 256) | 0 |
| batch_normalization_10 (Batc | (None, 256) | 1024 |
| dense_26 (Dense) | (None, 512) | 131584 |
| leaky_re_lu_19 (LeakyReLU) | (None, 512) | 0 |
| batch_normalization_11 (Batc | (None, 512) | 2048 |
| dense_27 (Dense) | (None, 1024) | 525312 |
| leaky_re_lu_20 (LeakyReLU) | (None, 1024) | 0 |
| batch_normalization_12 (Batc | (None, 1024) | 4096 |
| dense_28 (Dense) | (None, 30) | 30750 |
| reshape_4 (Reshape) | (None, 1, 30, 1) | 0 |

Total params: 720,670
Trainable params: 717,086
Non-trainable params: 3,584

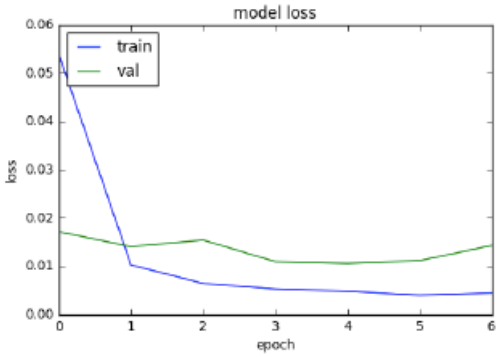
## 5. Experimentation

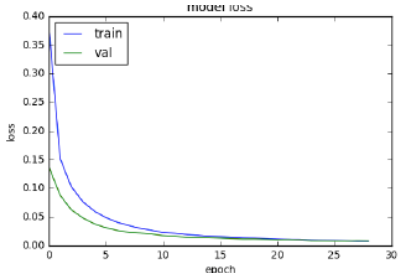
In our experimentation, I searched for which hyperparameters yielded the best accuracy. Specifically Since, Neural Networks are stochastic and output different results for each run with the same data. I have run the nodel for a specific number of iterations so that the average score of all the repetitions can determine the accuracy of the model. F-score is used as the accuracy metric (Optimisation metric) The F-score stabilised after around 60 iterations in the 2 hidden layer model. Hence that is the number of iterations, I use. Three different learning rates, 1E-03, 1E-02 and 1E-04 were tested for all cases. In addition to this, for each model, I tried unfreezing 0, 1, or 2 of the last layers. Finally, each model was trained and tested both without weight decay and with a weight decay of = 1E-05. I also experimented with 60 and 300 neurons fully dense layers (300 beter accuracy), 1 versus 2 layer MLP (2 layers performed better) and also different mini batch sizes (512/1024).



## 6.Results & Discussions

Here I present the results of my experimentation. Initially, I observed that there is high variance between the Training dataset and Validation. The binary cross entropy loss below is observed using Adam optimisation and learning rate alpjha of 0.001. To mitigate this, I employed a droput with keep prob = 0.5

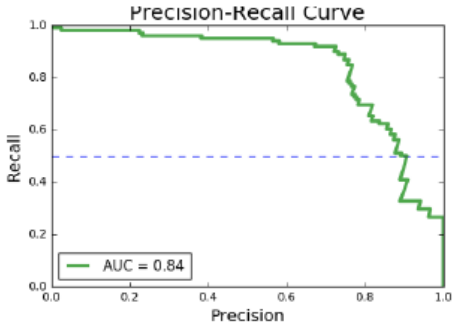



```
Model1_drop.add(Dropout(0.5)) Model1_drop.add(Dense(1, kernel_ini
tializer='he_normal', activation='sigmoid'))
```

model loss

Precision/Recall of 2 layer MLP (with RELU activation) with drop out is below - Recall_score: 0.9693877551020408, Precision_score: 0.6934306569343066 F-score: 0.8085106382978723

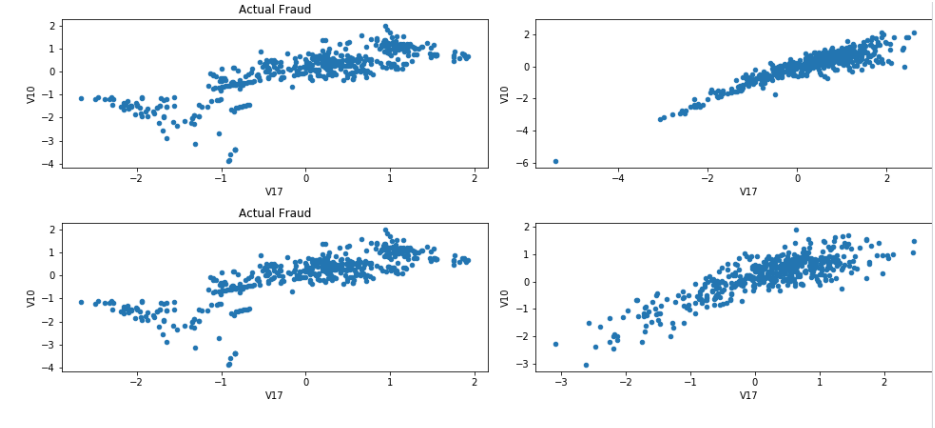



Performance of CNN (1D, 29 Kernel, 256 filters) averaged over multiple runs -F score, 0.860

```
Classification Report:

[Precision, Recall, F1, Support]
================================================================================================
==
0:      0.999718584009          0.999601374087      0.999659975612      85293
1:      0.7875          0.84        0.812903225806      150
================================================================================================
==
F1 Score, Fraud Class = 0.812903225806
```

GAN related losses (V17 versus V10 variable pattern) on actual fraud versus generated



## 6.1 Model Comparison

I trained the GANs using a training dataset that consists of all 492 fraudulent transactions. And can see that the actual fraud data and the generated fraud data through 5000 rounds of training. We can see the actual fraud data divided into the 2 KMeans classes, plotted with the 2 dimensions that best discriminate these two classes (features V10 and V17 from the PCA transformed features). We can see that the original GAN architecture starts to learn the shape and range of the actual data.

These GANs with generated and actual fraud were then trained using CNN and the average F1 scores of CNN are compared to MLP (2 layers with drop out, 1 layer) and Random Forest

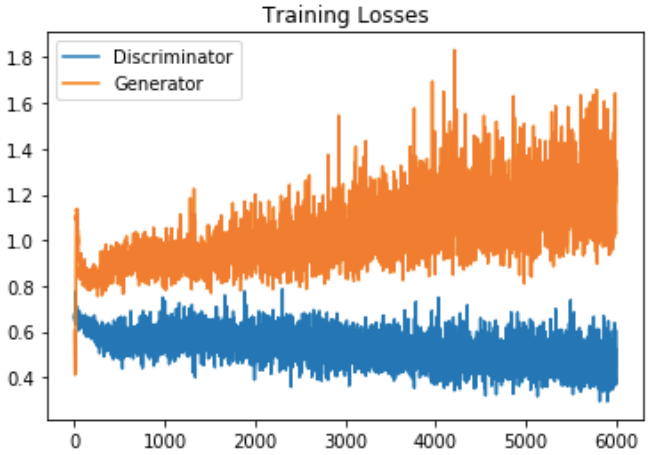
***CNN (best performance): 0.860230099502***

***RandomForestClassifier:0.846437 (baseline model)***

***MLPClassifier (2 Layer with drop out): 0.8085106382978723***

***MLPClassifier (1 Layer): 0.707243346007604***

GAN training losses of Generated and Discriminated are also shown below




## 6.2 Model Robustness and Model Analysis

CNN with enhanced fraud dataset through GAN yielded high training and test accuracy on our training and test sets. Adding drop out certainly yielded lower variance and additional layers in MLP enhanced performance. The poor generalization is a result of our dataset not being diverse. The training examples are from similar fraud patterns (around mid night , online data and a certain V10/V17 range). This could be fixed by having a more diverse training set or additional methods of dataset augmentation. Secondarily, we could have applied regularization methods that added more noise to help support generalization. In terms of hyperparameter tuning, a smaller learning rate yielded lower test accuracy for MLP and CNN. As the number of unfrozen layers increased, MLP was more prone to overfit to the train data. However, there was no clear impact on CNN's performance, probably due to its already higher accuracy with zero unfrozen layers and its larger number of layers. By the end of 5000 training iterations the generated fraud imaged pattern started to mimic actual fraud

## 7 Future Work

We can see that the GAN architecture starts to learn the shape and range of the actual fraud data pattern, but then collapses towards a small distribution. The generator has learned a small range of data that the discriminator has a hard time detecting as fake. There could be better architecture to ensure this mode collapse doesn't happen. Having augmented datasets can also help us minimise variance and lastly the network structure of CNN can be determined by data features, and the optimal permutation order of all permutations is determined by the feedback of model results. A feature sequencing layer can also be added to the CNN layer (to decide the order of input features)

## References


1. S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick, “Credit Card Fraud Detection Using Bayesian and Neural Networks,” in *Proceedings of the 1st International Naiso Congress on Neuro Fuzzy Technologies*, pp. 261–270, 2002.

2. K. Fu, D. Cheng, Y. Tu, and L. Zhang, “Credit Card Fraud Detection Using Convolutional Neural Networks,” in *Neural Information Processing*, vol. 9949 of *Lecture Notes in Computer Science*, pp. 483–490, Springer International Publishing, 2016.

3. S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, “Data mining for credit card fraud: a comparative study,” *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.

4. N. Carneiro, G. Figueira, and M. Costa, “A data mining based system for credit-card fraud detection in e-tail,” *Decision Support Systems*, vol. 95, pp. 91–101, 2017.

5. W. Yin, K. Kann, M. Yu, and H. Schtze, “Comparative Study of Cnn and Rnn for Natural Language Processing,” 2017, https://arxiv.org/abs/1702.01923.

6. R. Brause, T. Langsdorf, and M. Hepp, “Neural data mining for credit card fraud detection,” in *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '99)*, pp. 103–106, November 1999.

7. H. Shin, H. R. Roth, M. Gao et al., “Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

8. A. Correa Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, “Feature engineering strategies for credit card fraud detection,” *Expert Systems with Applications*, vol. 51, pp. 134–142, 2016.