ECSE 552 Project - GAN for Credit Card Fraud Detection

Amal Koodoruth, Aldo Abou Chedid, Banuso Abdulquadri, Yujing Liu
Group 2

***Abstract*—Credit card fraud detection is currently one of the most common problems in the world. This is due to the increase in both on-line transactions and e-Commerce platforms. Credit card fraud generally occurs when the card was stolen for any of the unauthorized purposes or when the fraudster uses the credit card information for use.**
Anomaly detection (AD), also known as, outlier detection or novelty detection, is referred to as the process of detecting data instances that significantly deviate from the majority of data instances, which follow some kind of distribution. Credit card fraud detection is an example of an anomalous event that occurs infrequently; therefore, there are not enough samples of positive fraud cases to generate a balanced dataset to effectively model an algorithm. Current credit card detection methods usually use the idea of classification, requiring a balanced training dataset that should contain both positive and negative samples. However, we often get highly skewed datasets with very few frauds.
Motivated by this, we propose the development of a Generative Adversarial Network (GAN), to detect fraud anomalies for a credit fraud detection dataset. Baseline algorithms, multilayer perceptron (MLP), convolutional neural network (CNN), and variational autoencoders (VAE), were developed as benchmark models. The results of these algorithms were analyzed based on accuracy, precision, recall, and F1 score. The experimental results show that our solution using GAN performs very well in comparison to the baseline models.

I. Introduction

Credit cards are used as a crucial payment method in modern society, and more fraudulent transactions are increasingly produced in the overwhelming of credit card usages. Fraudulent transactions affect not only banks and merchants, but also end users because even if they get reimbursement, they could eventually pay more for a higher fee of credit card services.
With the rapid advancement of deep learning, the demand for data is increasing. However, there are still many imbalanced datasets that are difficult to handle conventionally, which has attracted more and more attention in the academia and industry. For example, credit card fraud detection is a typical imbalanced classification problem. Using existing methods to classify balanced data can generally achieve good classification performance. However, the data in the real world is often imbalanced. If these methods are used to classify imbalanced data, the performance of the classifier will be degraded. Therefore, it is important to study the classification method for handling imbalanced datasets.
At present, the relevant methods of imbalanced data classification are mainly studied on the data preprocessing. Oversampling and undersampling are generally used. Oversampling is the most commonly used method to sample imbalanced data. The basic idea is to eliminate the data imbalance by increasing the size of the minor class. The undersampling method improves the classification performance of the minor class by reducing the size of the major class. The simplest method is to reduce the size of the major class by randomly removing some of the samples from the major class. The disadvantage is that some important information from the major class will be lost so that it cannot make full use of the existing information.
In this project, a classification method based on the generative adversarial network (GAN) is proposed for the classification of imbalanced data. GAN consists of a generative model and a discriminative model, and the two models are trained in a competitive way to achieve fast loss minimization. Competition between the generative model and the discriminative model leads to better discrimination.

II. Literature Review

We did a brief literature review on the state-of-the-art methods used in the industry to detect credit card fraud, revealed different methods to detect fraud in a credit card data set. Some existing methods are briefly broken down into supervised and unsupervised learning methods below:

A. Supervised Learning

Some anomaly detection problems for credit card fraud detection are solved using the classical supervised classification framework. In this way,a classifier is trained based on training data together with annotations, then classifies test transaction data into normal and abnormal categories. 4 supoervised learning methods for fraud detection are briefly explored below:

1) Logistic Regression: Logistic regression was developed by statistician David Cox in 1958 [4] and is a regression model where the response variable Y is categorical. Logistic regression allows us to estimate the probability of a categorical response based on one or more predictor variables x. It allows one to say that the presence of a predictor increases (or decreases) the probability of a given outcome by a specific percentage. Mathematically, logistic regression estimates a multiple linear regression function defined as:

$$Y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + .. + \beta_p x_{i,p} \tag{1}$$

where $x_{i,j}$ refers to the j^{th} predictor variable for the i^{th} observation, Y_i is the output of the i^{th} observation.

2) Decision Tree: Decision trees are simple but intuitive models that utilize a top-down approach in which the root node creates binary splits until a certain criteria is met. This binary splitting of nodes provides a predicted value based on the interior nodes leading to the terminal (final) nodes. In a classification context, a decision tree will output a predicted target class for each terminal node produced. Decision trees tend to have high variance when they utilize different training and test sets of the same data, since they tend to overfit on training data. This leads to poor performance on unseen data. Unfortunately, this limits the usage of decision trees in predictive modeling. However, using ensemble methods, we can create models that utilize underlying decision trees as a foundation for producing powerful results. [6]

3) Random Forest: The random forest algorithm, proposed by L. Breiman in 2001, [3] has been successful as a general-purpose classification and regression method. The approach, which combines several randomized decision trees and aggregates their predictions by averaging, has shown excellent performance in the setting where the number of variables is much larger than the number of observations. Moreover, it is versatile enough to be applied to large-scale problems, is easily adapted to various ad-hoc learning tasks, and returns measures of variable importance. In the classification context, the random forest classifier m is obtained via a majority vote among *K* classification trees with input x, that is,

$$m(x:\theta_1,....\theta_k) = \begin{cases} 1 \;\; if \;\; \frac{1}{K}\sum_{j=1}^{K} m(x:\theta_j) > \frac{1}{2} \\ 0 \;\; otherwise \end{cases} \tag{2}$$

where θ is the parameter set.

4) Extereme Gradient Boosting: Gradient boosting is a powerful machine learning technique for regression, classification and ranking problems, which produces a prediction model in the form of an ensemble of weak prediction models like decision trees. The model is built in a stage-wise manner. In each stage, it introduces a new weak learner to compensate the shortcomings of the existing weak learners. XGB stands for eXtreme Gradient Boosting, one of the implementations of gradient boosting concept. The unique of XGB is that it uses a more regularized model formalization to control over-fitting and achieves better performance. Gradient boosting relies on regression trees, where the optimization step works to reduce mean square error, and for binary classification the standard log loss is used. For a multi-class classification problem, the objective function is to optimize the cross entropy loss. Combining the loss function with a regularization term arrives at the objective function. The regularization term controls the complexity and reduces the risk of over-fitting. XGB uses gradient descent for optimization to improve the predictive accuracy at each optimization step by following the negative of the gradient as we are trying to find the sink in a n-dimensional plane. To learn the set of functions used in the model, XGB minimizes the following regularized objective

$$L(\theta) = \sum_{j=1}^{K} l(y_i, \hat{y_i}) + \omega(\theta) \tag{3}$$

where, θ is the learned parameter set, l is a differentiable convex loss function that measures the difference between the predictions $\hat{y_i}$ and the target y_i and ω is the regurlarization term.

B. Unsupervised Learning

There is a recent surge of interest in developing unsupervised generative models for anomaly detection. Generative models are trained to model the distribution of the normal transaction data distribution. Any transaction that does not follow the distribution is considered to be anomalous. In such a way, the fraud transaction can be detected in an unsupervised manner.
In this Section, we briefly discuss 3 unsupervised machine learning approaches for credit card fraud detection.

1) One-Class Support Vector Machine: One-Class SVM (OCSVM) was proposed by scholkopf to identify novelty / anomaly in an unsupervised manner without labeled training data. The algorithm learns a soft boundary in order to embrace the normal data instances using the training set, and then, using the testing instance, it tunes itself to identify the abnormalities that fall outside the learned region [2]. Mathematically, OCSVM is formulated by the following optimization problem :

$$Minimize \;\; \Phi(w) = \frac{1}{2}w^T w + \frac{1}{vn}\sum_{i=1}^{n} \xi_i - \rho \tag{4}$$

subject to:

$$y_i(\langle w, \phi(x_i)\rangle + b) \geq \;\; \rho - \xi_i, \;\; i = 1, ..., n \tag{5}$$

$$\xi_i \geq \;\; 0, \;\; i = 1, ..., n \tag{6}$$

The parameter v sets an upper bound on the fraction of outliers and a lower bound on the number of training examples used as support vectors.

2) Auto-Encoder: An auto-encoder (AE) learns to map from input to output through a pair of encoding and decoding phases. The encoder maps from the input to hidden layer, the decoder maps from the hidden layers to the output layer to reconstruct the inputs. The hidden layers of the auto-encoder are low dimensional and nonlinear representation of the input data. The AE is formulated as follows

$$\hat{X} = D(E(X)) \tag{7}$$

where, *X* is the input data, *E* is an encoding map, *D* is a decoding map and $\hat{X}$ is the reconstructeed input data. The objective of the autoencoder is to approximate the distrubution of *X* as accurately as possible. In particular, an autoencoder can be viewed as a soltuon to the following optimization problem:

$$min_{D,E}||X - D(E(X))|| \tag{8}$$

where $||.||$ is usually a 2-norm. Complex distributions of X can be modelled using a deep auto encoder with multiple layers, which refers to multiple pairs of the encoders and decoders.

3) Generational Adversarial Networks: GAN is a generative model designed by Goodfellow in 2014. In a GAN setup, two differentiable functions (generator G and discriminator D), represented by neural networks, are competing and trained simultaneously, which eventually drive the generated samples to be indistinguishable from real data.
During the training of GAN, generator and discriminator compete with each other and are trained alternately. After the discriminator training is converged, it can better distinguish between real non-fraud samples and other samples, including generated samples and fraud samples. The generator can generate samples which cannot be distinguished from real non-fraud samples, so that the discriminator can accurately determine whether the samples obey the distribution of real non-fraud samples. Samples which do not obey the distribution the non-fraud samples will be identified as a fraud by the discriminator.
The generator has a strong fitting capability, and the discriminator has a strong discrimination capability. The generator fits the distribution of nonfraud samples, and the discriminator determines whether samples are real nonfraud samples.
The methodology used in developing our GAN model for credit card fraud detection is expressed in the following section.

III. Methodology

As described previously, a GAN is a generative model composed of a differentiable generator function G and a differentiable discriminator function D. The intuiton behind the GAN is to create a pair that will have a behaviour based on a theoretical game where the generator tries to generate samples $\mathbf{x}_g$ that would decieve the discriminator while the discriminator will try to differentiate between the real - provided - samples $\mathbf{x}_r$ and the generated samples.
The generated samples are produced from a randomly sampled noise vector $\mathbf{z}$. For this implementation, $\mathbf{z}$ is a standard normal random vector. Both the generator and discriminator pipelines can be summarized as

$$\text{Generator:} \qquad \mathbf{x}_g = \mathbf{G}\left(\mathbf{z};\boldsymbol{\theta}^{(g)}\right), \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I}), \tag{9}$$

$$\text{Discriminator:} \qquad \hat{y} = D\left(\mathbf{x};\boldsymbol{\theta}^{(d)}\right). \tag{10}$$

where, $\mathbf{x}$ is randomly chosen between a generated sample $\mathbf{x}_g$ or a real sample $\mathbf{x}_r$. $\mathbf{G}$ and D are implemented as Multilayer Perceptrons (MLP) with parameters $\boldsymbol{\theta}^{(g)}$ and $\boldsymbol{\theta}^{(d)}$ respectively. The loss function is as follows:

$$L = \max_{G} E_{z\sim p_z} \log D(G(z)) \tag{11}$$

$$L = \max_{D} E_{x\sim p_r} \log D(x) + E_{z\sim p(z)} \log(1 - D(G(z))) \tag{12}$$

Here p_z denotes the multivariate standard normal distribution from which $\mathbf{z}$ is sampled and p_r denotes the distribution of real samples.

1) Dataset imbalance: As the task pertains to anomaly detection, the dataset consists of two parts which are extremely uneven in the number of samples: the normal points greatly outnumber the anomalous ones. This imbalance in the dataset complicates the use of most common classifiers. Since undersampling may result in loss of data information, and oversampling may change the data distribution. A generator capable of fitting a majority of the sample distribution is needed, and a strong discriminator which can determine whether samples obey the distribution is needed.
The proposed method uses the advantages of generator and discriminator and the competition between generator and discriminator to get a better performance of classification on imbalanced dataset.

2) Threshold Selection: The discriminator that we trained ended up being a classifier that can tell if a sample point is representative of the true data distribution it was trained on or not. It can therefore be used for AD. We assume that the anomalous data samples will be less representative of our distribution, hence have lower probability output by the discriminator. We identify the probabilities that in the lowest 10^{th} percentile, and the highest of them is our threshold probability,p. All the samples having probability less than p are classified as anomalies.

3) Model: Our generator attempts to imitate the generative process of the real data. As the real is compiled from multiple features that have been reduced to 31 via PCA, our generator is formed by a sequence of widening layers. The first layers get progressively wider to generate a high number of features that would imitate the initial features of the dataset. The last layer would imitate the effect of the PCA itself by reducing the high number of features back to the original number of dimensions. The architecture is detailed in Figure 1.

IV. Experiment and Results

A. Dataset

We are using the dataset containing transactions made by credit cards collected by the Machine Learning Group of ULB (Université Libre de Bruxelles) and Worldline cooperatively. This dataset covers two days' worth of credit card transactions in Europe, including 284,315 legitimate transactions and 492 fraudulent ones. Due to the need to protect users' anonymity, the original data was converted to a numerical value via PCA, and we do not know what they signify. The dataset has 31 features, 28 of them are named from V1 to V28 while the other three are named as Time, Amount and Class respectively. Here, Time refers to seconds elapsed since the first transaction, Amount refers to the transactions' amount, and Class is the class of transactions (value 1 represents fraud and 0 otherwise).

To get a suitable normal dataset size for evaluating the performace if our model, we have trained it with 5000, 10000, 20000, 50000, 100000, 150000, 200000 and 284315 (total) normal samples. So our dataset consisted of n training samples

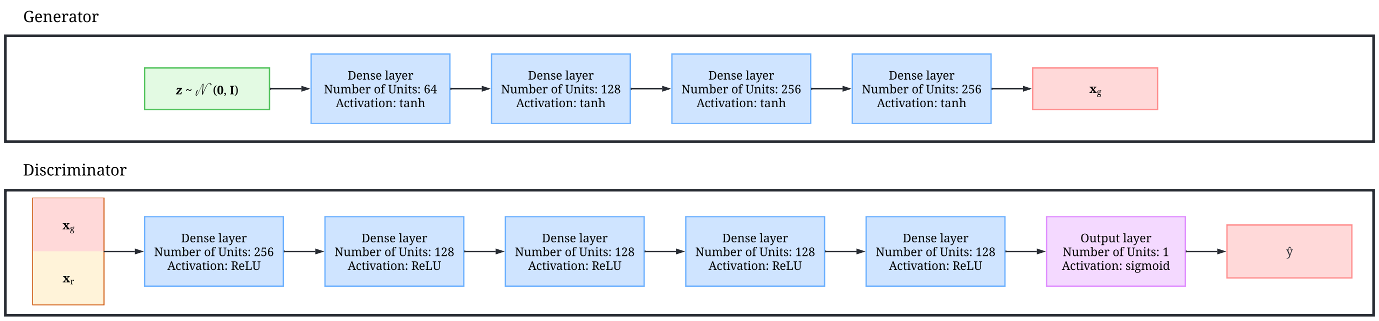
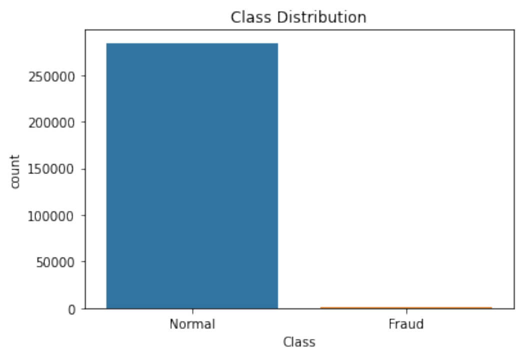

Generator
$z \sim \mathcal{N}(0, I)$
Dense layer Number of Units: 64 Activation: tanh
Dense layer Number of Units: 128 Activation: tanh
Dense layer Number of Units: 256 Activation: tanh
Dense layer Number of Units: 256 Activation: tanh
x_g
Discriminator
x_g
x_r
Dense layer Number of Units: 256 Activation: ReLU
Dense layer Number of Units: 128 Activation: ReLU
Dense layer Number of Units: 128 Activation: ReLU
Dense layer Number of Units: 128 Activation: ReLU
Dense layer Number of Units: 128 Activation: ReLU
Output layer Number of Units: 1 Activation: sigmoid
$\hat{y}$

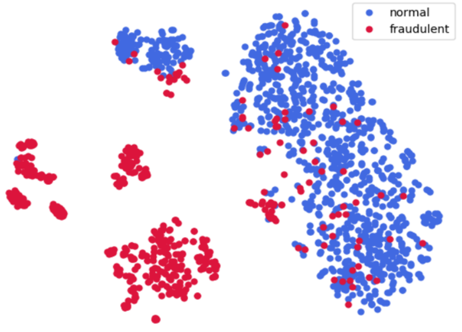
Fig. 1: GAN architecture

and 492 anomalous samples. We decided to use ⅔ of the dataset for training and ⅓ of it for testing.

Figure 4 summarizes the results. Very clearly, we can see that training with 5000 and 50000 normal samples give the best results on the test set. However, training with 5000 normal samples had a smaller runtime of 25s compared to 141s for training with 50000 normal samples. Also, the results obtained when training the model with 20000, 50000 and 100000 normal samples were, probably because of a combination of sampling and initialization, not reproducible. So we decided to perform our analysis with the model trained with the dataset containing 5000 normal samples.



(a)



(b)

Fig. 2: (a): Class Distribution , (b): TSNE

B. Evaluation Method

The aim was to develop a model that produces a high accuracy, good generalization with little optimization bias. We use accuracy, recall, precision, F1 score and AUPRC as the criterion for analyzing classification results and the calculation are as follow.

$$\text{Accuracy} = \frac{TP+TN}{P+N} \tag{13}$$

$$\text{Precision} = \frac{TP}{FP+TP} \tag{14}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{15}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP}+\text{FN})} \tag{16}$$

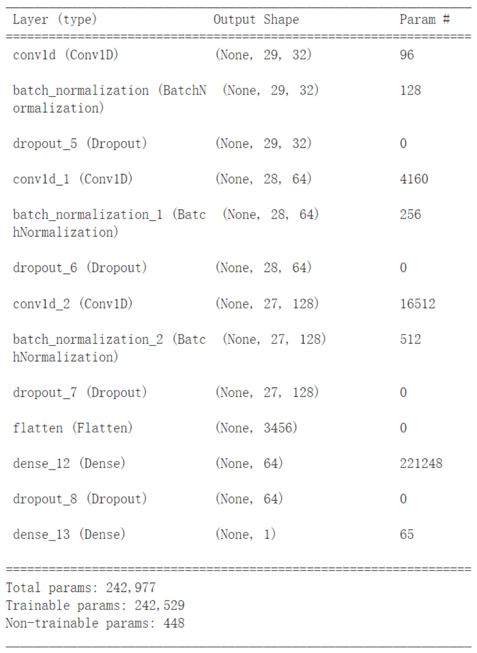
As previously mentioned above, the data set is highly imbalanced with 492 fraud records out of 284,807 transactions. Even all the samples are classified into non-fraud category, the classification accuracy is still extremely high, that means accuracy alone is not suitable as a primary performance objective. We use the Area Under the Precision Racal Curves (AUPRC) in our experimental study. A PR Curve is a plot of the precision (y-axis) and the recall (x-axis) for different probability thresholds. A skillful model is represented by a curve that bows towards a coordinate of (1,1). A no-skill classifier will be a horizontal line on the plot with a precision that is proportional to the number of positive examples in the dataset. A high area under the curve represents both high recall and high precision, high scores for both show that the classifier is returning accurate results, as well as returning a majority of all positive results.
We did not use the ROC for analysis as ROC plots could also be misleading when applied in imbalanced classification scenarios. In comparison PRC plots can provide the viewer with an accurate prediction of future classification performance due to the fact that they evaluate the fraction of true positives among positive predictions [7].

C. Baseline Comparision Results

We compare our model with 3 open sourse baseline model including MLP, CNN and VAE to analyze the performance. The models are tested on the dataset mentioned before with same dataset split as our GAN model. One extra step for adapting the baseline model is that all the three baseline methods need to delete the 'Time' class in advance to get a good anomaly detection result. The thresholds are set to the mean value of the anomaly data in training set.

1) MLP: The baseline MLP model [5] have 2 hidden layers of width 16 and 8. The activation function is ReLu and optimizer is Adam. The loss is sparse categorical cross entropy loss. After testing on the same dataset split described before, it could reach an accuracy of 98.12%, AUPRC score of 0.92 as shown in table I.

2) CNN: The structure of our CNN baseline [9] is shown in figure 3, it consists of 1D convolution layer, batch normalization, dropout layer, flatten layer and fully connected layer. The optimizer is Adam and the loss is binary cross entropy loss. After testing on the same dataset split described before it could reach an accuracy of 98.8%, AUPRC score of 0.95 as shown in table I.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 29, 32)            96

 batch_normalization (BatchN  (None, 29, 32)           128
 ormalization)

 dropout_5 (Dropout)         (None, 29, 32)            0

 conv1d_1 (Conv1D)           (None, 28, 64)            4160

 batch_normalization_1 (Batc  (None, 28, 64)           256
 hNormalization)

 dropout_6 (Dropout)         (None, 28, 64)            0

 conv1d_2 (Conv1D)           (None, 27, 128)           16512

 batch_normalization_2 (Batc  (None, 27, 128)          512
 hNormalization)

 dropout_7 (Dropout)         (None, 27, 128)           0

 flatten (Flatten)           (None, 3456)              0

 dense_12 (Dense)            (None, 64)                221248

 dropout_8 (Dropout)         (None, 64)                0

 dense_13 (Dense)            (None, 1)                 65

=================================================================
Total params: 242,977
Trainable params: 242,529
Non-trainable params: 448
_________________________________________________________________
```

Fig. 3: CNN Architecture

3) VAE: The baseline VAE model [10] has 4 layers with encoding dimension of 14 and 7. The activation function is ReLu, optimizer is Adam, and the loss is mean squared error loss. After testing on the same dataset split described before it could reach an accuracy of 88.2%, AUPRC score of 0.77 as shown in table I.

D. GAN Result

As descripted above, our GAN model structure is shown in figure 1. Since we are dealing with an unbalanced dataset, to reduce the class imbalance at data level approach, we perform an random under sampling on the nonfraudulent data. To check the effectiveness of undersampling method, we varies the normal sample size with 5000, 10000, 20000, 50000, 100000, 150000, 200000 and original size of 284315 unfraudulent data. We then feed the data to train the GAN model with enough epoch until the loss stabilize, after training we plot the AUC, AUPRC, recall, precision and F1-score of each model with corresponding size of the input data as shown in Figure 4.

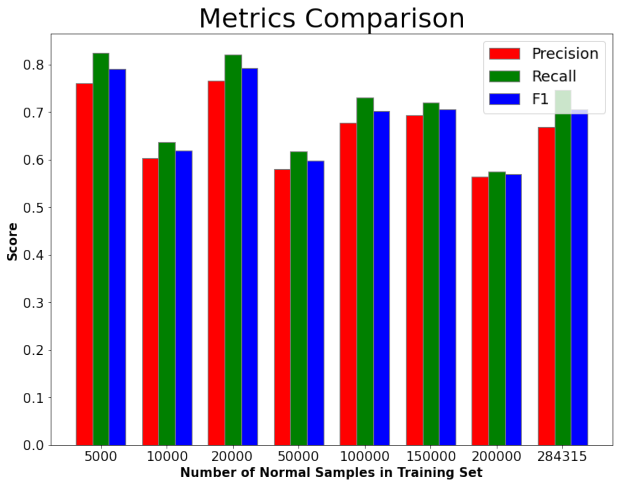


Fig. 4: Metrics Comparison

Model	AUC	AUPRC	Precison	Recall	F1-Score	Runtime
GAN	80%	0.85	0.76	0.82	0.79	25s
VAE	88.2%	0.77	0.41	**0.89**	0.56	30s
CNN	**98.8%**	**0.95**	**1.0**	0.87	**0.93**	86s
MLP	98.12%	0.92	0.94	0.85	0.89	**6s**

TABLE I: Model Performance

From Table I, we can see that the CNN outperforms our model in every accuracy metric. However, our model still gives decent accuracy results, which coupled with slower runtime, might be useful in some cases where quick training is more important than accuracy. Moreover, the recall is comparable in both models. This implies that there are few fraudulent transactions that that are incorrectly identified as normal, in both cases. Finally, we should not overlook the fact that our model is an unsupervised learning one, compared to the supervised CNN.

Comparing the GAN with the VAE, both unsupervised learning models, we can see that our model outperforms the VAE in precision, but the VAE has a better recall. This means that the VAE is better in identifying frauds, but previledges quantity too much. In other words, it flags a lot of transactions, and by doing so, it flags more frauds.

V. Conclusion and Future Work

A. Discussions and Conclusion

From the comparison metrics we could see that our GAN model has a relatively good performance with 5000 samples.

As the model takes more input samples the performance shows an unstable change. Comparing with the baseline model, our model's anomaly detection process in general have less runtime to achieve a good precision and recall value and AUPRC. The AUPRC and precision performance is better than the unsupervised method VAE, other performance in general is close to the other two supervised method. However, the VAE outperforms our model in recall, which means that VAE is better at identifying frauds. However, because of its very low precision, we can deduce that it does so just by flagging many transactions, which *hopefully* contain some frauds. This, in most cases, is not a major issue, but it can be frustrating for users to be declined transaction often. That being said, to combat credit card fraud, one can argue that it is a necessary evil and in that case, the VAE outperforms the GAN.

B. Future Work

We could look into more methods on handling the unbalanced dataset for future work. Right now most anomaly detection model are developed to solve the computer vision problem, we could look into the famous architectures including AnoGAN [8] and GANomaly [1] and use it as baseline model in solving our credit card fraud detection problem.

References

[1] Akcay, S., Abarghouei, A. A., and Breckon, T. P. Ganomaly: Semi-supervised anomaly detection via adversarial training. *CoRR abs/1805.06725* (2018).

[2] Bounsiar, A., and Madden, M. G. One-class support vector machines revisited. In *2014 International Conference on Information Science Applications (ICISA)* (2014), pp. 1–4.

[3] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[4] Cox, D. R. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological) 20*, 2 (1958), 215–242.

[5] Farahani, M. anomaly detection using ml, 2021.

[6] Navada, A., Ansari, A. N., Patil, S., and Sonkamble, B. A. Overview of use of decision tree algorithms in machine learning. In *2011 IEEE Control and System Graduate Research Colloquium* (2011), pp. 37–42.

[7] Saito, T., and Rehmsmeier, M. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one 10*, 3 (2015), e0118432.

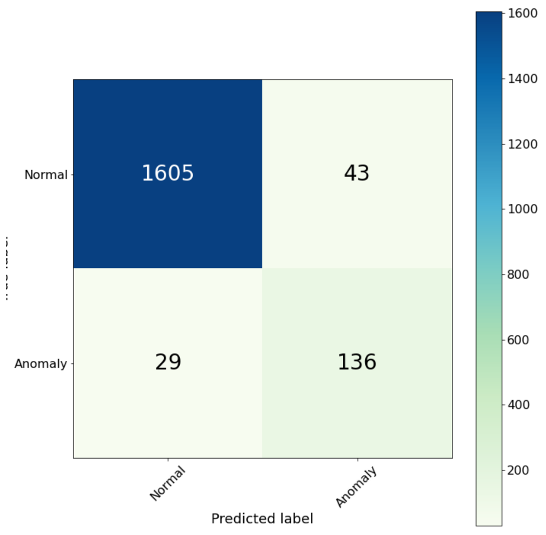
[8] Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U. M., and Langs, G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *IPMI* (2017).

[9] Sharma, E. Bank card fraud detection - unsupervised anomaly detection, 2020.

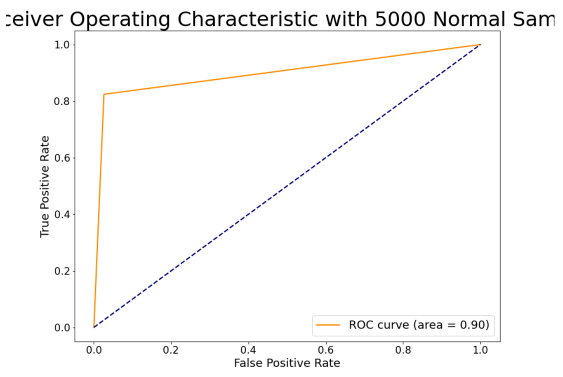
[10] Valkov, V. Credit card fraud detection using autoencoders in keras, 2019.

VI. Appendix

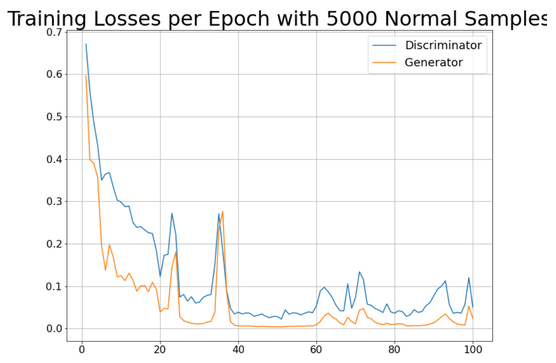
Appendix showing the different performance plots for sample distributions.



(a)

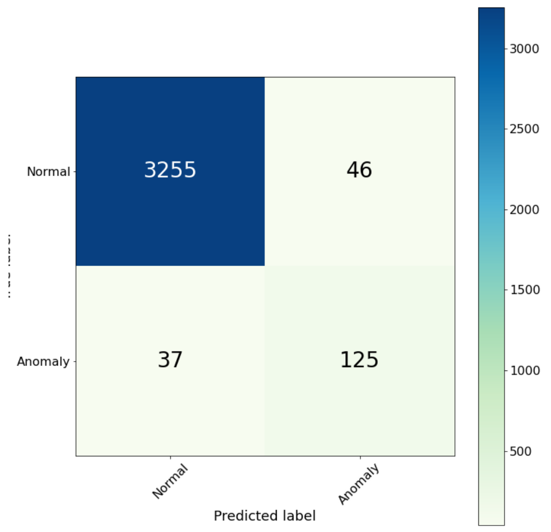


(b)

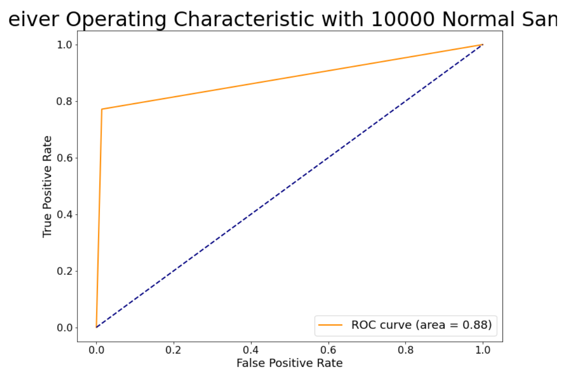


(c)

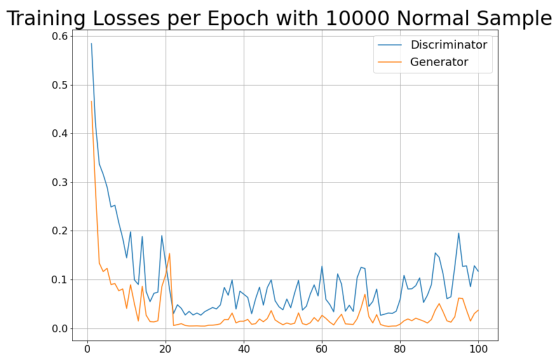
Fig. 5: (a): Confusion Matrix for 5000 samples, (b): Reciever Operating Characteristic for 5000 samples, (c): Training Losses for 5000 samples



(a)

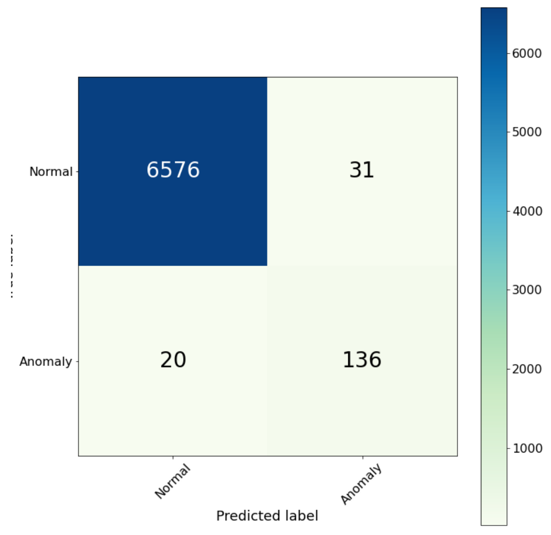


(b)

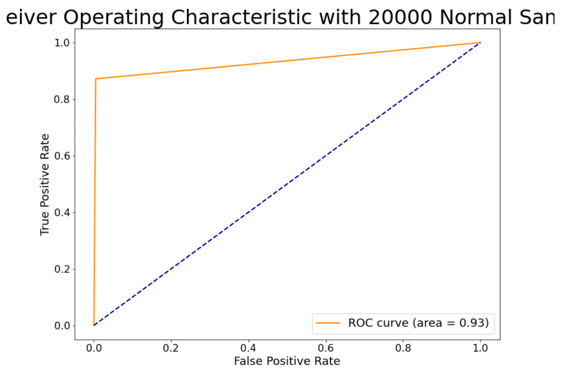


(c)

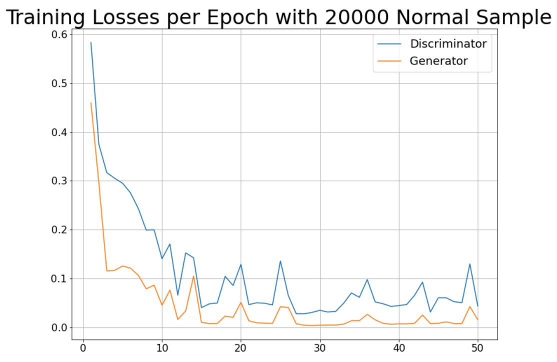
Fig. 6: (a): Confusion Matrix for 10000 samples, (b): Reciever Operating Characteristic for 10000 samples, (c): Training Losses for 10000 samples



(a)

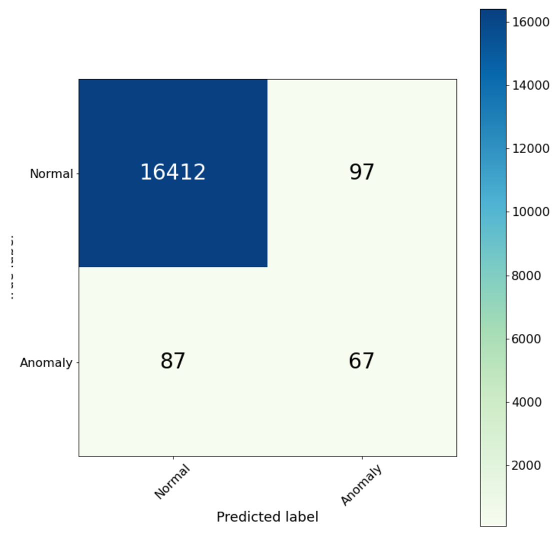


(b)

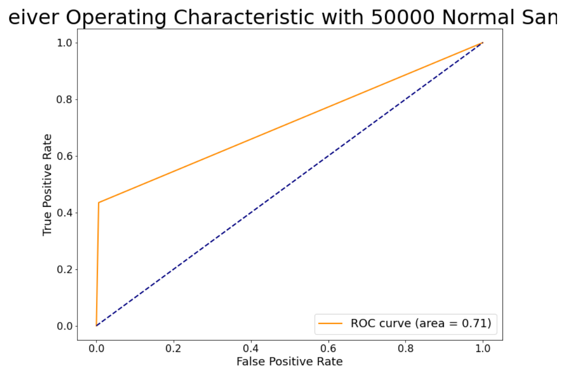


(c)

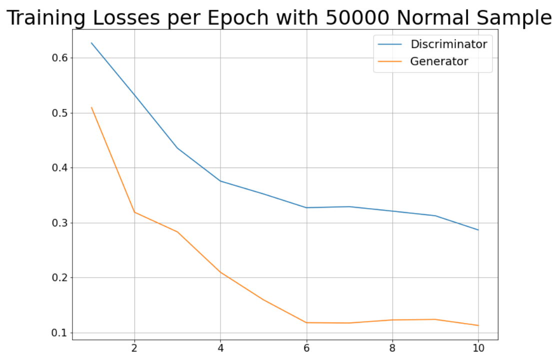
Fig. 7: (a): Confusion Matrix for 20000 samples, (b): Reciever Operating Characteristic for 20000 samples, (c): Training Losses for 20000 samples



(a)

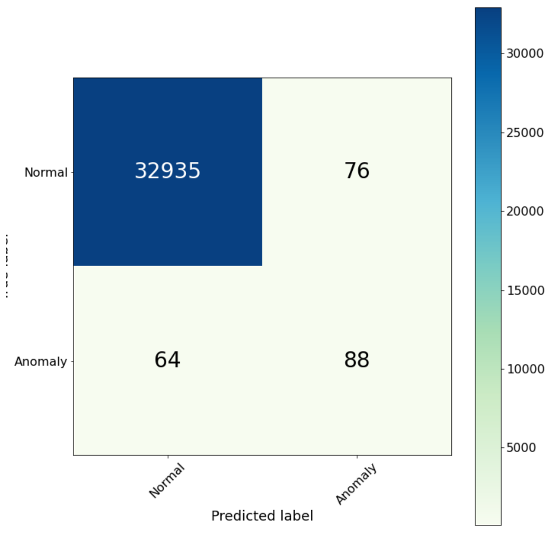


(b)

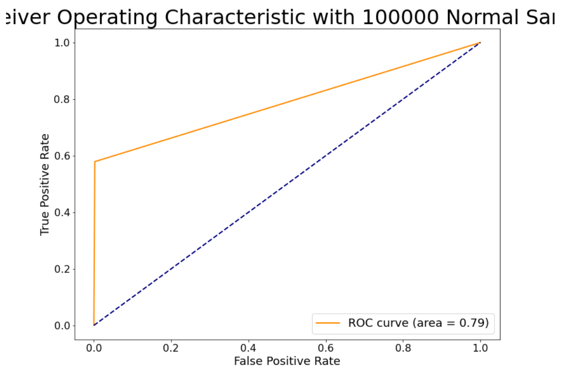


(c)

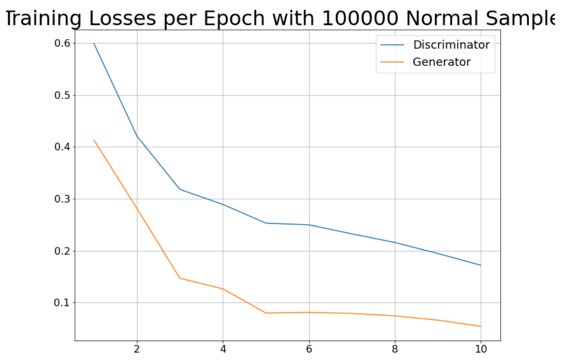
Fig. 8: (a): Confusion Matrix for, (b): Reciever Operating Characteristic, (c): Training Losses



(a)

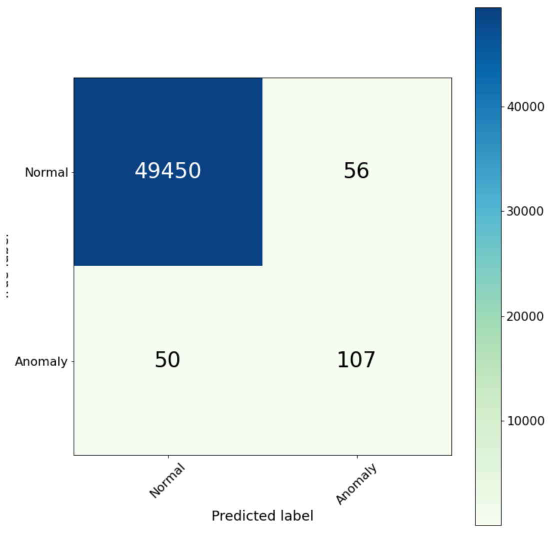


(b)

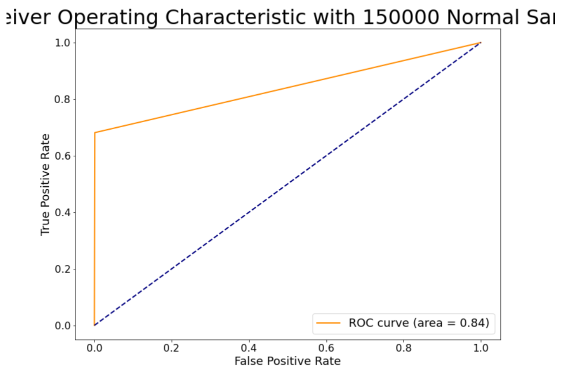


(c)

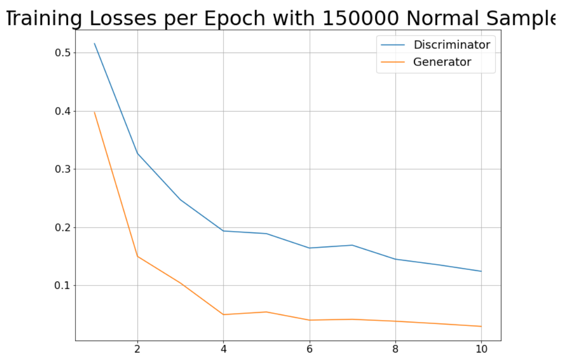
Fig. 9: (a): Confusion Matrix for 100000 samples, (b): Reciever Operating Characteristic for 100000 samples, (c): Training Losses for 100000 samples



(a)

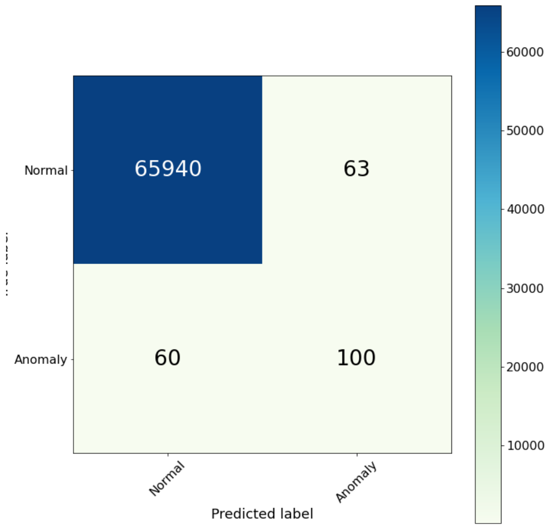


(b)

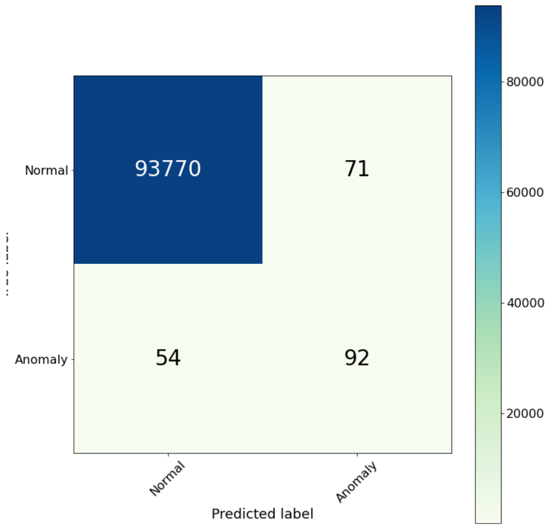


(c)

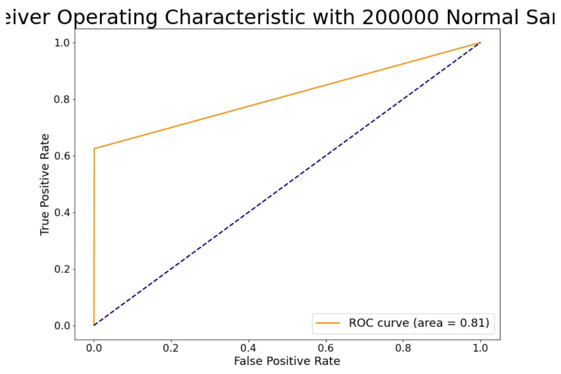
Fig. 10: (a): Confusion Matrix for 150000 samples, (b): Reciever Operating Characteristic for 150000 samples, (c): Training Losses for for 150000 samples



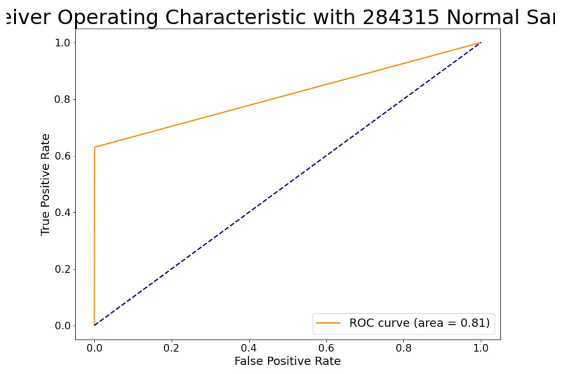
(a)



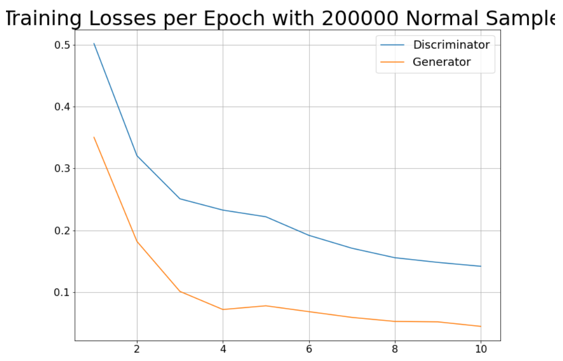
(a)



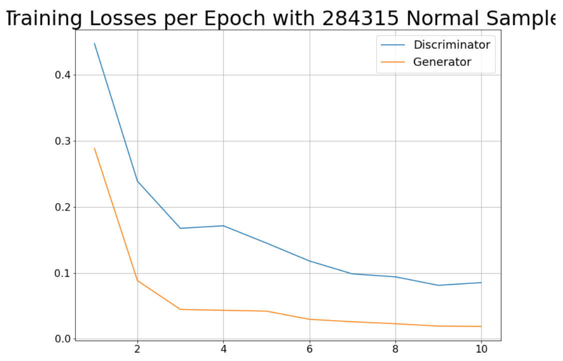
(b)



(b)



(c)



(c)

Fig. 11: (a): Confusion Matrix for 200000 samples, (b): Reciever Operating Characteristic for 200000 samples, (c): Training Losses for 200000 samples

Fig. 12: (a): Confusion Matrix for 284315 samples, (b): Reciever Operating Characteristic for 284315 samples, (c): Training Losses for 284315 samples