COSE474-2022F: Final Project
Anomaly Detection in Finance: Credit Card Fraud Detection

Sung-Yoon Jo

1. Introduction

Anomalies are data that have signifcant different characteristics from the majority of normal data. Many recent studies have focused on proposing efficient anomaly detection methods.[17] Among many areas of interest, Credit Card Fraud Detection(CCFD) is at center of attention in Finance. With the advancement of modern technology, large volumes of credit card transaction data are produced at rapid rates. These data are characterized by high-dimensions and large number of samples, making it difficult for quick identification of various fraud patterns.[3]

1.1. Motivation

Recently Deep Learning (DL) methods have been proven to be very effective in processing high-dimentional data such as image, text, videos, and audio. In particular, deep neural networks (DNN) have gained advantage due to their iterative and instance-wise training methods.[1] However, DNN peformance in CCFD still fails to measure up to the state-of-the-art ML methods such as XGBoost or Random Forest. We believe that with appropriate hyperparameter tuning, DL methods can be just as effective as the traditional ML methods, and also gain the upper hand in tasks of **incremental learning** and **imbalanced data handling**.

Through this paper, we'd like to propose two anomaly detection frameworks based on DL: i) A feed-forward deep neural network(DNN) for Supervised Learning, and ii) An Autoencoder implemented with SMOTE for Unsupervised Learning. The DNN will be compared with the traditional ML methods in the hope of achieving just a successful performance. The Autoencoder will be compared to the SOTA methods presented on the source code website: DevNet[13] and Deep Iso Forest[18].

Contributions can be described as following:

- Implement incremental learning through design of a fully connected feed-forward DNN.
- Address the imbalance of data issue by design of Autoencoder applied with the SMOTE technique.

2. Related Works

2.1. Traditional ML Based Methods

The most successful models for fraud detection are gradient boosting algorithms such as XGBoost [5], and tree-based models such as Random Forest [4]. Other ML methods such as LightGBM, one-class SVM, or Logistic Regression are also popular. With the right preprocessing and feature extraction, these models will produce high performance results.

2.2. Deep Anomaly Detection

Many studies have been conducted that attempt to apply the DL based methods in CCFD. (Sarkar,2021)[15] created XB-Net, an extremely boosted neural network that attempted to combine tree-based models with neural networks to achieve greater robustness. (Kazemi et al.,2017)[11] used Autoencoder and achieved an accuracy of 81.6%.

3. Method

3.1. Challenges

One main challenge is the inherent **class imbalacnce** present in the data. In general, transaction data contain far less fradulent ones. The percentage of fradulent transactions is well under 1%. To deal with the large difference between classes, additional learning strategies of oversampling through SMOTE will be implemented for imbalanced learning.

The choice of performance metrics is another issue to be addressed. For imbalanced data, common metrics like Accuracy cannot be implemented. For DNN, we will try to use other metrics such as Precision or Recall.[7] For Autoencoder, to compare with the SOTA's AUC ROC metric we will try to deal with the data imbalance issue first hand.

Careful selection of optimal hyperparameters is crucial for best results. We will conduct grid search to find such hyperparameters.

3.2. Proposed Framework with Hyperparameter Setting

The model for the neural network is the Sequential from the keras module.[10] The input layer will take the features of the credit card transactions, and the output layer will return a single value indicating the binary category of the transaction(1 or 0). The number of neurons in each hidden layer decreases from 256 to 1 sequentially as we go down the network. Between hidden layers the ReLU activation function was used. The output layer activation function was sigmoid since this is a binary classification problem.[6] To prevent overfitting, Dropout with probability of 0.3 was conducted.[16] Batch Normalization was applied to ensure a quick and stable training process. As for the optimizer, Adam was selected because it ensures decent performance without much tuning. However, we lowered the default learning rate to 1e-4 because the training process became very unstable. The loss function was binary crossentropy since this is a binary classification problem.

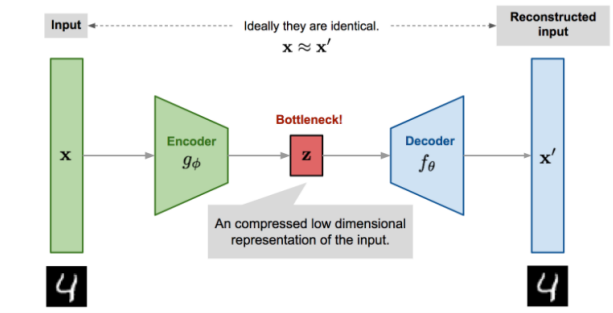

Figure 1. Autoencoder Architecture

The autoencoder is a special type of model that uses solely the descriptive features. The goal is to learn representations to reconstruct the original data in a most accurate fashion. By comparing the original data with the reconstructed one, we can derive the reconstruction error which will be an indicator for fraud risk.[2] We implemented a fully connected bottleneck design of input-encoder-latent-decoder-ouput(3 hidden layers). Each layer has 29,14,7,7,29 neurons, respectively. The dimension of the output of the decoder is the same as that of the input of the encoder. L1 regularization was applied to adjust for the decreasing number of features as we go down the encoder network. To ensure fast convergence and big learning steps, activation function "tanh" was implemented along with "relu".

The process of Autoencoder design is written in the following pseudocode:

Algorithm 1 Autoencoder Pseudo Code

Input: dataset X, parameters (w,b_x,b_h), where b_x is the Encoder parameter and b_h is the Decoder parameter.
Initial Variables: h: a vector for hidden layer, $\hat{X}$: reconstructed X, L: vector for Loss Function, l: batch number.
$h, \bar{x}, L, l \longleftarrow null, i = 0$
while $i < 1$ **do**
$h = f(p[i] * w + p[i] * b_x)$

$$\hat{X} = g(p[i] * w^T)$$

$$L = sum(X - \hat{X})^2$$

$\theta[i] = {}^{min}_{p} L(X - \hat{X})$
Return θ
end while

4. Experiments

For model design, training, and assessing performance, computing resource of graphical processing units(GPU) was used along with modules of tensorflow and keras from google colab.

4.1. Dataset

The dataset contains transactions from credit cards in September 2013 by European cardholders. Out of the 284807 transactions, exactly 492 are frauds. Thus only 0.172% of all the transactions are fradulent, resulting in high imbalance of data. The features V1,V2,...,V28 are PCA transformed ones. Features 'Time', 'Amount', and 'Class(1 or 0)' are also included. Data is normalized through the MinMaxScaler. To avoid overfitting, early stopping strategy and validation process are implemented. SMOTE [9] technique is used to deal with data imbalance. The minority(fraud) class will be oversampled by generating examples in the neighborhood of observed ones. This resampling technique will be beneficial to AUC ROC.

4.2. Competing Methods

Two SOTA methods are presented on the source code website: i) DevNet and ii) Deep Iso Forest. DevNet is an end-to-end anomaly score learning framework that combines the method of gaussian prior-based referenece scores and Z-score based Deviation Loss.[13] Deep Iso Forest is a framework with an ensemble of representations derived from DNN, and operates simple axis-parallel isolation to build iTrees in the forest.[18] DevNet and Deep Iso Forest achieved an AUC-ROC performance score of 0.98 and 0.953, respectively.

4.3. Performance Evaluation

For DNN, due to the imbalanced nature of data it would be inappropriate to use accuracy from the Confusion Matrix to assess performance. Instead we use metrics of Precision, Recall, and F1-Score to compare with other ML methods.[7] For Autoencoder, after data balancing through SMOTE we use the metric of AUC ROC.[9] This metric can be obtained by plotting the True Positive Rate against the False Positive Rate for all different classification thresholds to create the ROC curve.[8] The area under the curve is then calculated. AUC ROC will be used for comparison purposes with other SOTA models.

4.4. Results

First we display the result of DNN. With the dataset being large, a relatively large batch size was set through trial and error. Prior small mini-batch gradient descent values of 32,64,128,... proved to be very unstable and displayed overfitting. To sufficiently minimize the error, an optimal high epoch values of 10000 was set. After training, the loss was graphed as following:

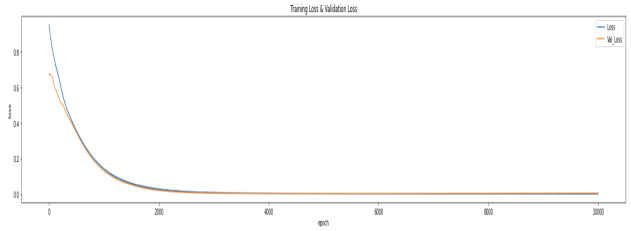

Figure 2. Loss Function vs Number of epochs of DNN

A final loss of 0.0014 was displayed in the training data and 0.0031 for the test data. As for the performance metrics, the final values were precision:0.81, and recall:0.95.

The following table displays comparison of DNN with other ML models. A threshold value of 0.5 was used.

	Precision	Recall	F1_Score
DNN	0.81	0.95	0.87
XGboost	0.84	0.94	0.89
RandomForest	0.82	0.96	0.88
LGBM	0.10	0.07	0.08
SVM	0.82	0.84	0.83

Table 1. Results of Performance Metrics on DNN vs ML methods

Result: In terms of F1_Score, DNN could not outperform the SOTA of XGBoost and Random Forest. However, there is only a slight difference in performance.

As for autoencoder, a batch size of 256 and an epoch of 30 proved to be appropriate through grid search. The ROC curve after training was graphed as following:

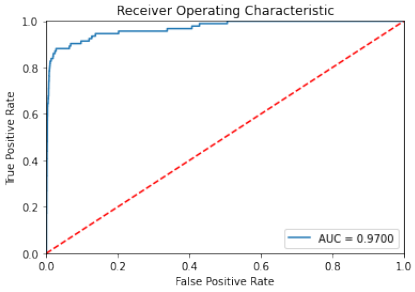

Figure 3. ROC Curve of Autoencoder

An AUC_ROC value of 0.97 was achieved.
The following chart displays comparison of AUC_ROC values between Autoencoder and SOTA methods: DevNet and DeepIsoForest.

	AUC ROC
Autoencoder	0.97
DevNet	0.98
DeepIsoForest	0.953

Table 2. Results of Performance Metrics on Autoencoder vs SOTA

Result: The Autoencoder could not outperform SOTA of DevNet, but had better performance than the runner-up: DeepIsoForest. Again, there is only a slight difference in performance.

5. Conclusion

Compared to classical methods, DL methods can be time-consuming as there are an infinite set of hyperparameter possibilities. Indeed in this project alone, it was not easy to find optimal hyperparameters via grid searching. Although the two DL methods failed to outperform SOTA models, there was not a great difference. Greater expressitvity and high-dimensional feature extraction was possible through the complex layers of neural network and Autoencoder. We conclude that our two DL models are just as competitive as the classical ML or the SOTA approaches, mixed with additional advantages of incremental learning and dealing with data imbalance. For future research, using more advanced models like CNN or LSTM could achieve SOTA through delicate modeling.[14] Rather than solely using AUC ROC, performance metrics such as Precision-Recall Curve or Average Precision could be more effective in dealing with the imbalance problem.[12] We hope this research was helpful in CCFD and paves the way for future work.

References

A, M. and D., C. Recent advances on effective and efficient deep learning-based solutions. *Neural Computing and Applications*, 34, 2022.

An, J. and Cho, S. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2:1–18, 2015.

Bank, E. C. 6th report on card fraud. URL https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport202008~521edb602b.en.html#toc2.

Breiman, L. Random forests. machine learning. *Springer*, 45, 2001.

Chen, T. and Guestrin., C. A scalable tree boosting system. *ACM SIGKDD International Conference*, 22, 2016.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, pp. 303–314, 1989.

Davis, J. and Goadrich, M. The relationship between precision-recall and roc curves. *23rd International Conference on Machine Learning*, pp. 233–240, 2006.

Fawcett, T. Roc graphs: notes and practical considerations for researchers. *Machine Learning*, 31:1–38, 2004.

Hall, N. V. C. K. W. B. L. O. and Kegelmeyer, W. P. Smote: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, 16:321–257, 2002.

Jayawardana, R. Analysis of optimizing neural networks and artificial intelligent models for guidance, control, and navigation systems. *researchgate*, 2021.

Kazemi, Z. and Zarrabi, H. Using deep networks for fraud detection in the credit card transactions. *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pp. 0630–0633, 2017.

Muschelli, J. Roc and auc with a binary predictor: a potentially misleading metric. *Journal of Classification*, pp. 1–13, 2019.

Pang, G., Chunhua, S., and van den, H. A. Deep anomaly detection with deviation networks. *arXiv*, 2019.

Pankaj, M., Anusha, R., Gaurangi, A., Lovekesh, V., Puneet, A., and Gautam, S. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv*, pp. 1–5, 2016.

Sarkar, T. Xbnet: An extremely boosted neural network. *arXiv*, 2021.

Sutskever, N. S. G. H. A. K. I. and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15:1929–1958, 2014.

Thudumu S., Branch, P. J. J. A comprehensive survey of anomaly detection techniques for high-dimensional big data. *Journal of Big Data*, 7, 2020.

Xu, H., Guansong, P., Yijie, W., and Yongjun, W. Deep isolation forest for anomaly detection. *arXiv*, 2022.

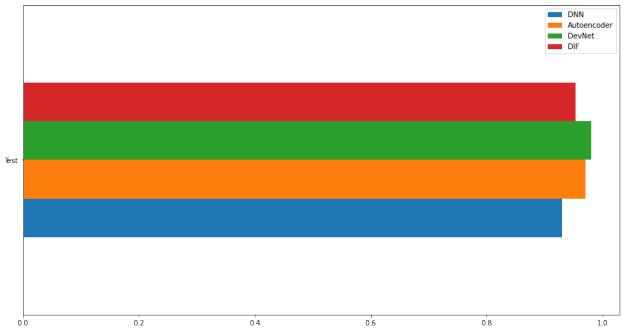

Figure 4. Barplot of AUC ROC of Autoencoder vs DevNet vs DeepIsoForest

Appendix

click [here] for Github Code Address

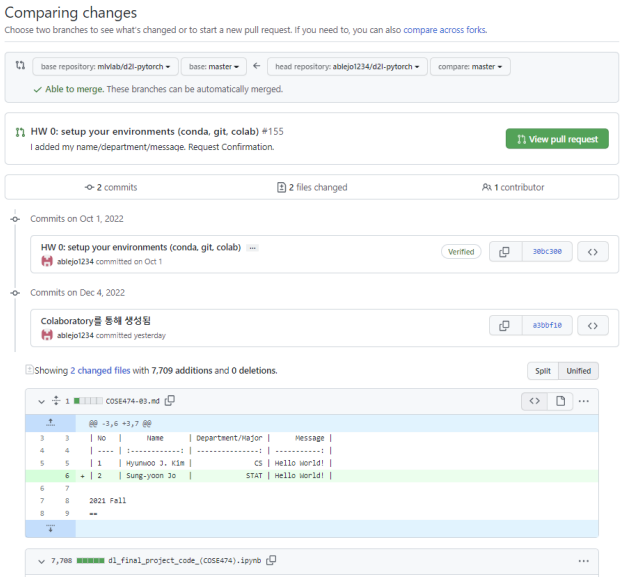

Figure 5. Screenshot of Commit History