Anomaly Detection for Credit Card Fraud Detection

Mannal Kamble, Karthvik Sarvade, Ajay Kumar Nagilla
New York University – Tandon School of Engineering
mk8475@nyu.edu, ks6807@nyu.edu, na3370@nyu.edu
mannalkamble/Anomaly-Detection-for-Credit-Card-Fraud-Detection (github.com)

Abstract

In this project, we aim to reproduce the results of the research paper titled "Locally Interpretable One-Class Anomaly Detection for Credit Card Fraud Detection" by Tung-Yu Wu and You-Ting Wang, which developed a novel deep learning-based anomaly detection framework for credit card fraud detection. This framework addresses the challenge of imbalanced transaction datasets. Our replication effort focuses on reconstructing the paper's proposed dual-component model, consisting of an AutoEncoder for learning and reconstructing legitimate transaction patterns, and a classifier for detecting deviations that signify fraudulent activities. The goal is to replicate the architecture accurately and achieve performance levels comparable to those reported in the paper. This endeavor is critical for validating the model's effectiveness and practicality in real-world scenarios and establishes a benchmark for the reproducibility of sophisticated AI models in the domain of financial fraud detection.

Introduction

In the digital finance era, the prevalence of credit card fraud poses a significant challenge, with a direct impact on consumers and financial institutions worldwide. The increase in online financial activities has led to a parallel rise in fraudulent transactions, highlighting the urgent need for more effective fraud detection mechanisms. This project report introduces a deep learning model designed to enhance the detection of fraudulent credit card transactions, a crucial step towards bolstering cybersecurity in financial transactions.

The primary challenge in credit card fraud detection is the skewed nature of transaction data, where legitimate activities significantly outnumber fraudulent ones. Conventional detection methods often struggle with this imbalance, leading to inefficiencies in identifying rare, yet impactful, fraudulent cases. To tackle this, our project employs an advanced deep learning approach, inspired by "Locally Interpretable One-Class Anomaly Detection for Credit Card Fraud Detection" by Tung-Yu Wu and You-Ting Wang. Our model innovatively combines an AutoEncoder for accurate reconstruction of transaction data and a fully-connected classifier, adept at discerning complex patterns in imbalanced datasets typical of credit card transactions.

In this report, we delve into our endeavor to meticulously reproduce the results of the advanced deep learning model originally proposed for credit card fraud detection. The document meticulously outlines our experimental approach, detailing the setup, execution, and analysis methods used to validate the model's effectiveness. By closely replicating the study and rigorously testing it against Kaggle credit card fraud detection dataset, we aim to not only confirm the model's robustness but also to explore its practical applicability in real-world scenarios. This effort is instrumental in affirming the reliability and relevance of the model in the ongoing battle against credit card fraud, providing a solid foundation for future research and development in the field of financial cybersecurity.

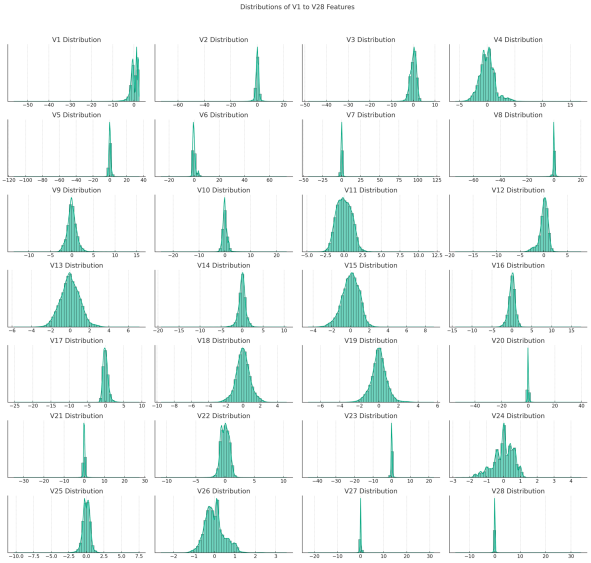

Figure 1 - Distribution of features in dataset

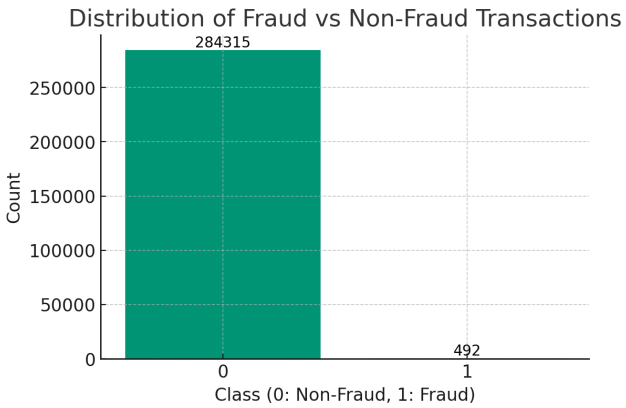

Figure 2 - Class imbalance in the dataset

Literature Review

The challenge of credit card fraud detection, a critical area in the financial sector, has grown significantly with the rise of digital transactions. Detecting fraudulent activities in highly imbalanced datasets, where legitimate transactions vastly outnumber fraudulent ones, is a key difficulty. The evolving nature of fraud tactics further complicates this task. Studies like those by Abbas et al. (2021) and Thanh Thi Nguyen et al. (2020) emphasize the necessity of sophisticated methods to efficiently identify rare fraudulent activities without compromising the identification of legitimate transactions.

Anomaly detection has been identified as a promising approach to address these challenges. It focuses on identifying patterns that deviate significantly from normal transaction behavior. Traditional anomaly detection methods, including statistical models, clustering techniques, and nearest neighbor methods, often struggle with the complexity and dynamic nature of credit card data.

The integration of deep learning into fraud detection marks a significant advancement in this field. Deep learning techniques, capable of processing large volumes of data and uncovering subtle patterns, have shown considerable promise. AutoEncoders, a form of neural network, are particularly effective. They excel in reconstructing inputs and identifying anomalies through reconstruction errors, making them suitable for fraud detection in imbalanced datasets, as demonstrated in research by Pumsirirat and Yan (2018) and Wu and Wang (2021).

Further advancements include the use of adversarial networks and ensemble methods to detect sophisticated fraud schemes more effectively. For example, the use of Generative Adversarial Networks (GANs) for data augmentation, as explored by Ba (2019), addresses class imbalance in fraud detection datasets. Additionally, the emerging field of explainable AI, as discussed by Barredo Arrieta et al. (2020), is making deep learning models in fraud detection more transparent and trustworthy.

The literature indicates a trend towards more advanced, accurate, and adaptable models for credit card fraud detection. Future research is likely to focus on enhancing real-time detection capabilities, addressing the challenges posed by evolving fraud tactics, and improving model interpretability to meet the stringent demands of financial institutions and regulatory bodies.

Architecture

Autoencoder - Generator:

Encoder:
1. Composition: Begins with a linear layer reducing the feature space from 28 to 15, followed by Batch Normalization and a ReLU activation. This process is then repeated to compress the representation further to 8 dimensions.
2. Functionality: This encoding process distills the essential information from the input, capturing the key features in a more compact form.

Decoder:
1. Reverse Architecture: The decoder mirrors the encoder's structure but in reverse order, reconstructing the data back to its original feature space. It starts by expanding the 8-dimensional encoded data back to 15 dimensions, followed by a similar sequence of Batch Normalization and ReLU activation, and finally returns to the original 28-dimensional space.
2. Reconstruction Goal: The decoder aims to reconstruct the input data as closely as possible, which is crucial for the Autoencoder's role in anomaly detection.
Forward Pass: The model takes input data, compresses it via the encoder, and then attempts to reconstruct the original data through the decoder. The quality of reconstruction serves as an indicator of how well the model has learned to represent normal transaction patterns.

Fully Connected Neural Network - Discriminator:
Layer Configuration:

1. Input Layer: With 28 neurons, corresponding to the 28 features of the transaction data.

2. Intermediate Layers: A sequence of layers, starting with 10 neurons, includes Batch Normalization to stabilize learning and normalize the input for each layer. The ReLU activation function introduces non-linearity, allowing the model to learn complex patterns.
3. Output Layer: A single neuron to output the classification result. The network is designed to output a binary classification indicating whether a transaction is fraudulent.

Sequential Processing: The architecture follows a sequential processing pattern where the input passes through each layer in order, culminating in the final classification output.

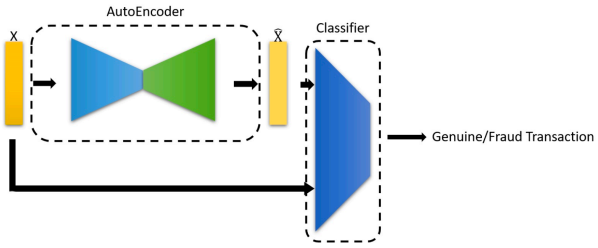

Figure 3 - Model Architecture Diagram

Training and Optimization:

Parameter Configuration:

1. Batch Size: Set to 4096, determining the number of samples processed before the model updates its parameters.
2. Learning Rate: Configured at 2e-4, which controls the step size at each iteration while moving toward a minimum of the loss function.
3. Number of Epochs: The training process runs for 1 epoch, where an epoch is one complete pass through the entire training dataset.
4. Normalization: Utilizes Z-score normalization to standardize the input data, ensuring that each feature contributes proportionally to the final prediction.

Optimizers: Separate Adam optimizers with weight decay (1e-4) are defined for both models. Adam optimizer is known for its efficiency in handling sparse gradients and its adaptability with respect to the learning rate.

Loss Functions:
1. Reconstruction Loss: The SmoothL1Loss is used for the Autoencoder. This loss function is less sensitive to outliers than the Mean Squared Error loss and generally provides better convergence for regression tasks.
2. Adversarial Loss: Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss) and Mean Squared Error Loss (MSELoss) are employed for the FCNN. BCEWithLogitsLoss combines a sigmoid layer and the BCELoss in one single class, which is more numerically stable than using a plain Sigmoid followed by a BCELoss.

Methodology

1. Dataset Preparation and Splitting:

Custom Dataset Class 'SplitedDataSet':
Purpose: To segregate credit card transactions into two categories: fraud and non-fraud.
Data Loading: Reads data from a CSV file, ensuring all transactions are loaded correctly.
Class Filtering**:** Depending on the mode ('non-fraud' or 'fraud'), the dataset filters out the opposite class of transactions to create a focused dataset.
Data Conversion**:** Each transaction's features and labels are converted from string format to float for numerical processing.

Aggregated Dataset 'DataSet':
Function: To combine multiple subsets of data into a single dataset.
Data Aggregation: Appends features and labels from different datasets, ensuring that the combined dataset contains a balanced representation of both fraudulent and non-fraudulent transactions.
Torch Conversion: Features and labels are converted to FloatTensors for compatibility with PyTorch models.

Dataset Splitting 'getDatasets':
Data Splitting: Splits the non-fraudulent and fraudulent data into training and testing subsets.
Training and Testing Proportions: Determines the number of data points for each subset, ensuring a representative sample for both training and evaluation.
Random Splitting: Utilizes random_split for a randomized division of data, enhancing the model's generalizability.

2. Neural Network Architecture:

Weight Initialization Function:
Goal: To optimize the initial weights of the neural network layers.
Layer-specific Initialization: Applies different initialization strategies for convolutional, linear, and BatchNorm1d layers to ensure efficient learning.

FCNN (Discriminator):
Architecture: Comprises linear layers, batch normalization, and ReLU activations in a sequential manner.
Role: To classify the input transactions into binary classes (fraudulent or non-fraudulent).

Autoencoder (Generator):

Encoder and Decoder: The encoder compresses the input data, and the decoder reconstructs it, aiming to capture and reproduce the underlying patterns of the transactions.
Purpose: To learn the normal pattern of transactions and assist in anomaly detection.

3. Model Training and Evaluation Setup

Setting Training Parameters:
Batch Size: Set to 4096, determining the number of samples processed in one iteration.
Learning Rate (lr**)**: A value of 2e-4 is chosen, balancing the speed and stability of the training process.
Epochs: The number of complete passes through the entire dataset is set to 1 for initial training.
Normalization: Z-score normalization is used to standardize features, ensuring uniform data scaling.
Reconstruction Loss Type: Specified as 'SmoothL1', a combination of L1 and L2 losses, which is less sensitive to outliers.

Initialization of Models:
Autoencoder (Generator) and FCNN (Discriminator) are initialized and set to training mode, enabling specific behaviors like dropout during the training phase.

Defining Optimizers:
Adam Optimizers: Separate optimizers for the generator and discriminator, with weight decay set to 1e-4 to help prevent overfitting.

Loss Function Setup:
Reconstruction Loss: nn.SmoothL1Loss() is chosen for the generator, measuring the difference between reconstructed and original data.
Adversarial Loss Functions: Binary Cross-Entropy (nn.BCEWithLogitsLoss()) and Mean Squared Error (nn.MSELoss()) for the discriminator.

4. Training Loop
Iterating Over Epochs:
The loop runs for the specified number of epochs, processing the training data in batches.

Generator Training:
Reconstruction: The generator creates a reconstructed version of the features.
Loss Calculation: Reconstruction loss (SmoothL1Loss) and Binary Cross-Entropy loss are computed based on the reconstructed and real data.
Backpropagation and Weight Update: The generator's weights are updated using its optimizer to minimize the combined loss.

Discriminator Training:
Real and Fake Predictions: The discriminator makes predictions on both real and reconstructed (fake) data.
Losses for real and fake predictions are computed separately using Binary Cross-Entropy.
Backpropagation and Weight Update: The discriminator's weights are updated to improve its ability to distinguish real from fake data.

5. Testing and Performance Evaluation Loop

Setting Models to Evaluation Mode:
The generator and discriminator are switched to evaluation mode, ensuring dropout and batch normalization layers behave correctly during testing.

Iterating Over Different Thresholds:
The loop tests the model's performance at varying thresholds to determine the best balance for fraud detection.

Processing Test Data:
For each batch in the test dataset, the generator reconstructs the features, and the discriminator predicts the probability of fraud.

Calculating Confusion Matrix Elements:
True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) are calculated based on the discriminator's predictions and actual labels.

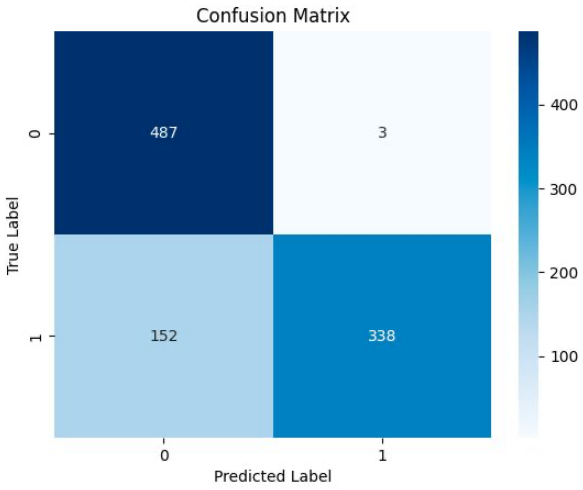

Figure 4 - Confusion matrix

Computing Performance Metrics:
Key metrics such as accuracy, recall, precision, F1 score, and Matthews Correlation Coefficient (MCC) are calculated and displayed.

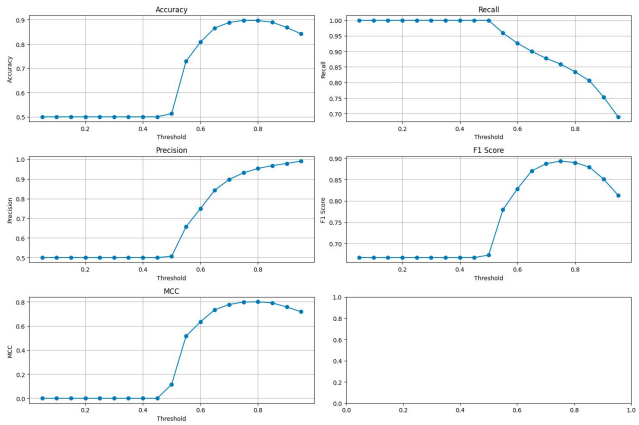

Figure 5 – Performance metrics at different thresholds

ROC Curve and AUC Visualization:
The Receiver Operating Characteristic (ROC) curve is plotted, and the Area Under the Curve (AUC) is calculated for visual analysis of the model's performance.

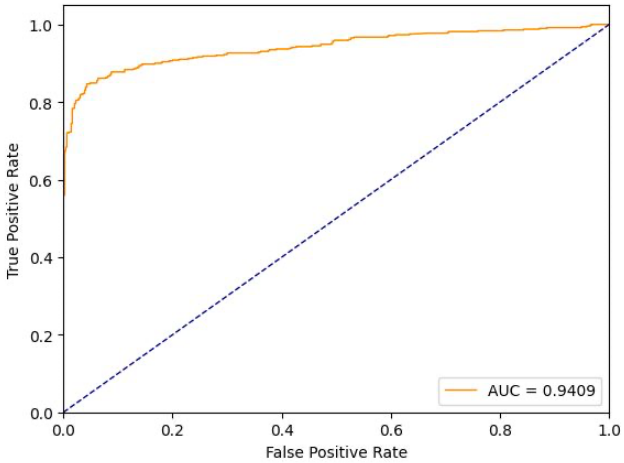

Figure 6 - ROC Curve

Results

In our credit card fraud detection study, the model exhibited promising results, especially when compared to the research paper benchmarks. We observed a peak performance at specific thresholds, with the model striking a balance between precision and recall. The most notable achievement was at a threshold of 0.8, where the Matthews Correlation Coefficient (MCC) reached 0.8001, closely approaching the research paper's MCC of 0.8128. This result is particularly significant as MCC is a comprehensive metric that takes into account true and false positives and negatives, indicating a robust model performance in balanced classification of fraudulent transactions.

Metric	Our bench-marks	Research Paper Bench-marks
Accuracy	0.8969	0.9061
Precision	0.9534	0.9216
Recall	0.8347	0.8878
F1-score	0.8901	0.9044
MCC	0.8001	0.8128
AUC	0.9409	0.9434

Table 1- Comparing performance metrics

Conclusion

In our project, we set out to replicate the deep learning-based anomaly detection framework from the research paper "Locally Interpretable One-Class Anomaly Detection for Credit Card Fraud Detection" by Tung-Yu Wu and You-Ting Wang. We successfully reproduced the model and methodologies described in the paper.
Beyond the scope of our report, we embarked on experimental endeavors to test the model's applicability to different datasets. These experiments, though challenging, revealed crucial insights into the factors impacting anomaly detection accuracy. Moreover, our exploratory work with adversarial attacks and subsequent model retraining provided valuable perspectives on the model's resilience. This phase of our project, though not extensively documented in the report, significantly contributed to our understanding of the model's robustness and versatility in varying contexts. These additional experiments and findings are also available on our GitHub repository.

References

1. Abbas, S., & Nahavandi, S. (2021). Uncertainty-Aware Credit Card Fraud Detection Using Deep Learning. arXiv:2107.13508.
2. Ba, H. (2019). Improving Detection of Credit Card Fraudulent Transactions using Generative Adversarial Networks. arXiv:1907.03355 [cs.LG].
3. Barredo Arrieta, A., et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities, and challenges toward responsible AI. Information Fusion, 58, 82–115.
4. Pumsirirat, A., & Yan, L. (2018). Credit Card Fraud Detection Using Deep Learning Based on Auto-encoder and Restricted Boltzmann Machine. International Journal of Advanced Computer Science and Applications, 9(1), 18-25.
5. Thanh Thi Nguyen, M., Abdelrazek, M., & Babar, A. (2020). Deep Learning Methods for Credit Card Fraud Detection. arXiv:2012.03754.

6. Wu, T., & Wang, Y. (2021). Locally Interpretable One-Class Anomaly Detection for Credit Card Fraud Detection. arXiv:2108.02501 [cs.LG].

Acknowledgments

We thank Professor Gustavo Sandoval for his guidance, mentorship, and expertise throughout this project. We would also like to extend our gratitude to the teaching assistants of the course, whose support and assistance were instrumental in the successful completion of this project.