Credit Card Fraud Detection Using AutoEncoder Based Neural Networks

Shiming Jin
Wee Kim Wee School of Communication and Information
NTU
Singapore, Singapore
jins0009@e.ntu.edu.sg

***Abstract*—This research paper aims to train an autoencoder model for the purpose of identifying fraudulent credit card transactions. An Autoencoder model is used to reduce the reconstruction error for valid transactions while increasing the reconstruction error for fraudulent ones in order to detect them. The dataset used in this study contains 284,807 credit card transactions, of which 492 are fraudulent. The dataset is split into 80% training and 20% testing data, where the training data only contains valid transactions to make the model better at reconstructing valid transactions. The results of this study show that the autoencoder model achieved a recall rate of 86.99%, indicating that it can effectively identify fraudulent credit card transactions. The paper describes the general procedures of preprocessing, hyperparameter tuning, model building, model testing, and evaluation. Additionally, the challenges encountered during the process are discussed, along with the solutions to overcome them. Overall, this research provides a useful approach for identifying fraudulent credit card transactions, which can be beneficial for financial institutions and their customers.**

Keywords—Fraud Detection, Anomaly Detection, Autoencoder

I. Introduction

Credit card fraud is a significant problem affecting both individuals and businesses worldwide. According to recent statistics, losses due to credit card fraud amounted to approximately $28.6 billion in 2020 worldwide, and this number is projected to increase in the coming years. Despite the numerous measures taken to prevent credit card fraud, it remains a persistent challenge for financial institutions, governments, and individuals.

Detecting fraudulent credit card transactions is a challenging task due to the complex and dynamic nature of the transactions. Fraudsters are continuously developing new techniques and methods to bypass the existing fraud detection systems, making it difficult for traditional methods to detect and prevent fraudulent transactions. As a result, there is a need for new and more advanced techniques to identify fraudulent transactions accurately.

One such technique is using autoencoder models, which can identify fraudulent transactions by reducing the reconstruction error for valid transactions while increasing the reconstruction error for fraudulent ones. The trained model can then flag any transaction that exceeds a certain threshold as potentially fraudulent. Such a model would benefit not only banks but also other industries that rely on credit card transactions, such as e-commerce businesses, insurance companies, and government agencies.

The benefits of detecting fraudulent transactions are significant. For banks and financial institutions, it helps to prevent financial losses and maintain the trust of their customers. It also reduces the administrative and legal costs associated with resolving fraudulent transactions. Additionally, detecting fraudulent transactions helps to prevent identity theft, which is a growing concern for individuals and businesses alike. Therefore, developing an effective fraud detection system is critical to maintaining the integrity of the financial system and protecting consumers' assets.

II. Challenges

One of the major challenges in this project is the imbalanced dataset, with a significant difference in the number of fraudulent transactions compared to valid ones. The lack of adequate data for the minority class can lead to model overfitting. Traditional classification models can achieve high training accuracy indicating high performance in classifying valid transactions but cannot effectively learn to detect fraudulent transactions during testing since the model did not acquire enough information about the pattern of fraudulent transactions.

To address this challenge, the Autoencoder model is used. Autoencoders use an encoder and a decoder in the model for training. Valid transactions are passed into the encoder and decoder for training purposes. The encoder reads the original information, and the decoder tries to reconstruct the original data. The reconstruction error is the loss or difference between the reconstructed data and the original one. The weights within the artificial neural network are tuned through the training process in which the model is trained to be talented in identifying and reconstructing valid transactions. In the testing phase, where both valid and fraudulent data are given, the trained autoencoder can efficiently reconstruct valid transactions with minimal error while simultaneously detecting the fraudulent transactions due to their high reconstruction errors. This approach does not require records with minority labels to be present during the training phase which solves the data imbalance problem.

Another challenge faced in this project is the lack of a GPU machine, which can lead to longer training times and limit the number of experiments that can be run. Since my computer does not support GPU, thus the training time is long, especially with large datasets like credit card transaction data. To overcome this challenge, a simpler architecture instead of the full architecture is used initially. Moreover, fewer training samples were obtained. By further lowering the number of epochs, a simpler system for unit testing is constructed to make sure the system is working correctly before training on the entire dataset. This can help to reduce training time and enable more experiments to be run, leading to a more robust architecture with optimal performance.

Additionally, a high recall was required in this project to minimize false negatives and the associated costs for the

misclassification of fraudulent records. To achieve a high recall rate, the threshold of testing loss was lowered to mark more records as fraudulent. While this approach may result in lower precision, it will include most of the fraudulent data and reduce the cost of misclassification.

Furthermore, the large hyperparameter search space can pose a challenge during the hyperparameter tuning phase. This process can be time-consuming and computationally expensive. To address this challenge, Optuna, a package for hyperparameter optimization, was utilized. Optuna can automatically search for the best combination of hyperparameters for training in an efficient fashion for the autoencoder model. This optimal hyperparameter space searching technique can boost the quality of hyperparameter tuning and reduce computational costs and time by running fewer iterations.

Overall, using an Autoencoder model, starting with a simpler architecture, setting a lower threshold for testing loss, and utilizing Optuna for hyperparameter optimization can help to overcome the challenges of an imbalanced dataset, a lack of a GPU machine, high recall, and a large hyperparameter search space. By addressing these challenges, a more accurate and efficient model can be developed to detect fraudulent transactions, ultimately benefiting banks and other industries where credit card fraud is a concern.

III. METHODOLOGY

A. *Preprocessing*

Understanding the data is a crucial step in any data analysis project, and viewing the head of the data helps to get an overview of what the dataset contains.

The dataset was imported as a pandas data frame and the head of the data was viewed. It contained 284,807 credit card transactions with 31 columns. Most of the columns contained information about the transaction but were encoded to prevent data breaches and privacy leaks.

Removing irrelevant features is important in machine learning projects as it reduces the number of attributes of the dataset to prevent the curse of dimensionality. Moreover, it is easier to train the model and improve its performance when the dimension is reduced. In the current dataset, the 'Time' column was dropped since it was irrelevant to detecting fraudulent transactions. This feature did not contribute to determining whether the transaction was fraudulent or not but would only add noise to the training dataset.

Dealing with null values is an essential step in any data analysis project, as null values can significantly affect the performance of the model during training and testing. Thus, the data.isnull().sum() function helped in checking whether the dataset contained null terms. The result showed that it did not contain any null terms.

An imbalanced dataset is a common problem in machine learning projects and can lead to difficulties in model selection. To check whether this problem occurred, the ratio of valid transactions and fraudulent ones were checked. The result showed that there were only 492 out of 284315 records that were fraudulent, indicating that the dataset was extremely imbalanced. In this case, traditional classification models cannot be used for this task since the training process will not provide sufficient information for detecting fraudulent transactions, which are the minority. Instead, an anomaly detection task is required, where the model is trained to identify abnormal transactions that are potentially fraudulent. To visualize the class frequency, a histogram was plotted below, which showed a significant difference in the number of valid and fraudulent transactions.

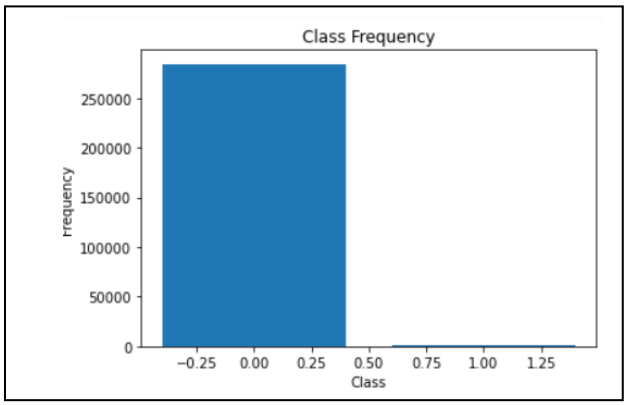


Fig. 1. Frequency Distribution of Classes

Splitting the dataset into training and testing sets is essential in machine learning projects as it helps to evaluate the model's performance on unseen data. Moreover, a validation set is essential as it is necessary for tuning the hyperparameters, such as learning rate, batch size, and loss function, which affect the model's performance during training. The credit card transactions dataset was separated into valid and fraudulent transaction records. The ratio was set to be 80% for training and 20% for testing. Out of the 80% for training, 80% of the training records were used to train each hyperparameter combination during the hyperparameter tuning phase. The rest 20% was used to test the performance of each hyperparameter combination.

Some attributes had obviously higher values than others, which could affect the weight of each attribute during training. Scaling the dataset helps to normalize the values of the features and makes them comparable to each other. Shuffling the dataset is necessary to randomize the order of the records, preventing the model from learning the order of the records instead of learning the patterns in the data. To apply these principles to the task at hand, the dataset was shuffled with random seed 77 and then scaled using StandardScaler() function. Using a random seed is necessary to ensure that the randomization process is the same for each run, allowing for reproducible results.

Finally, I converted the datasets to PyTorch tensors and separated the labels from the datasets. PyTorch tensors are the input format for training neural networks in PyTorch, and separating the labels from the datasets is necessary to use them as targets during training.

B. *Hyperparameter Tuning*

Optimizing the hyperparameters of a neural network model is essential for achieving optimal performance. One common method of hyperparameter tuning is using packages like Optuna. Optuna is a Python package that provides a framework for hyperparameter optimization by using Bayesian optimization algorithms. It enables users to easily define a search space of hyperparameters and allows for the optimization of a chosen objective function.

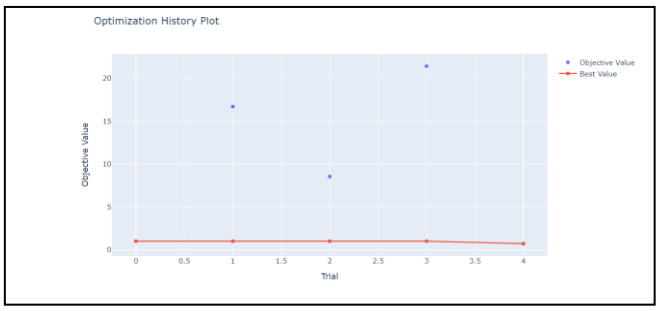
To implement the hyperparameter tuning procedure in this project, an Autoencoder model with an encoder and decoder was first constructed. The encoder and decoder both include multiple linear layers, batch normalization layers, dropout layers, and activation functions. The input and output of the model have the same shape as the original credit card transaction data. The forward function combines the encoder and decoder for the forward propagation of the neural network.

Next, a train_and_evaluate function() was implemented to perform the training and evaluation of the model. The loss function and optimizer were set to a testing value within the parameter space dictionary, while the batch size was set to a value within the same parameter space dictionary. The parameter space dictionary contains all the possible values for testing for each hyperparameter. The value assigned to the variables in the train and evaluation function were placeholders that refer to the parameter space dictionary. The model then implemented the forward and backward propagation step along with an update of weights after training each batch of training data. The evaluation mode was then activated, and the loss function's reduction parameter was set to 'none', allowing the use of reconstruction error for each record as a metric for determining the performance of each hyperparameter combination. The mean of all records' reconstruction loss was used as a metric for determining the performance of the hyperparameters. The hyperparameter combination with lower loss was chosen.

The objective() function was then implemented, which defined the parameter space for searching the best parameter combination and returned the loss. To run the hyperparameter tuning, optuna.create_study() was used with 'minimize' as the direction parameter, indicating the objective of this tuning task is to find a parameter combination that minimizes the mean loss. The TPESampler() was passed in as the second parameter, which performed better than GridSearchCV() and RandomizedSearchCV() since it recorded the hyperparameter selection with good performance and continued to use some of its hyperparameters for the next iteration with only small changes on several hyperparameters that contribute negatively towards the overall performance. Finally, the study was run through study.optimize(), with the objective function passed in as the first parameter, and 5 as the number of trials to reduce computation time.

The following diagrams show the history and performance of optimization of the hyperparameter combinations.

Fig. 2. Optimization History Plot



The Optimization History plot show that the objective value is lowest in trial 0 until trial 3. However, in trial 4, the best value is updated, providing an objective value of 0.713, lower than the objective value of trial 0, which is 1.003. Thus, the hyperparameter combination used in trial 4 is selected as the best hyperparameter combination.

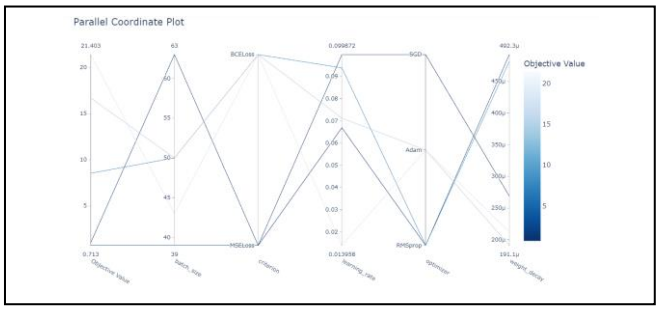


Fig. 3. Parallel Coordinate Plot

The Parallel Coordinate Plot not only shows the objective values of each trial but also shows the choice of hyperparameter combinations for each trial. This gave us a more direct view of the optimization process of the Optuna hyperparameter tuning. The best combination of hyperparameters was the following:

```
learning_rate: 0.09987188426582617
optimizer: SGD
criterion: MSELoss
batch_size: 39
weight_decay: 0.00026878723981259067
```

Fig. 4. Best Hyperparameter Combination

C. *Model Training and Testing*

Model training and testing are crucial steps in the development of an Autoencoder model for credit card fraud detection. To begin with the model training process, the best parameter combination obtained from hyperparameter tuning is used to set the hyperparameters of the model. Then, the model is defined and the loss function, optimizer, and other necessary parameters are set. The model is then trained using forward and backward propagation while continuously updating the model weights and the loss for each batch. The output shows that the loss has been reduced through epoch 1 to epoch 10, indicating an improvement in the model's ability to reconstruct valid transaction records.

```
Epoch:1, Batch:1, Loss:1.1972    Epoch:10, Batch:4662, Loss:0.7746
Epoch:1, Batch:2, Loss:1.0337    Epoch:10, Batch:4663, Loss:0.8493
Epoch:1, Batch:3, Loss:1.5794    Epoch:10, Batch:4664, Loss:0.8620
Epoch:1, Batch:4, Loss:0.9977    Epoch:10, Batch:4665, Loss:0.6575
Epoch:1, Batch:5, Loss:1.4841    Epoch:10, Batch:4666, Loss:0.5303
```

Fig. 5. Comparison of MSE Loss for batches in Epoch 1 and Epoch 10

Moving on to the model testing phase, the model is first changed to evaluation mode to ensure that the dropout and batch normalization layers are working properly during testing. Then, the gradient descent procedure is shut down to freeze the model weights, and the model is tested using the testing dataset. A scatterplot of MSE loss against record ID is created to visualize the distribution of the MSE loss among the records. It is observed that most of the data have an MSE loss lower than 50, indicating that the model is capable of reconstructing most of the records accurately. However, there are some records with higher MSE loss that are likely to be fraudulent records since the model is not good at reconstructing fraudulent records. Overall, the model testing phase provides valuable insights into the performance of the Autoencoder model and allows for further improvements to be made.

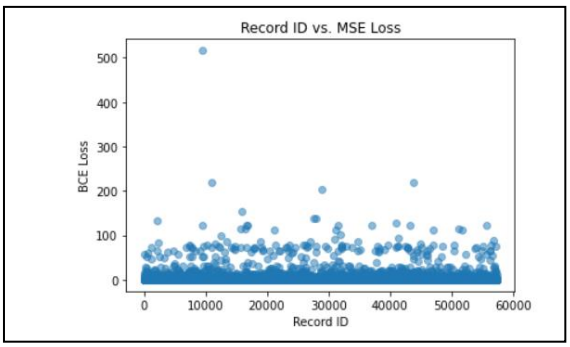


Fig. 6. MSE Loss as Reconstruction Error against Record ID

D. Model Evaluation

To evaluate the performance of the credit card fraud detection autoencoder model, several metrics were utilized, including a confusion matrix, precision-recall curve, and ROC curve. Each of these metrics provides unique insights into the model's performance and can be used to make informed decisions about the model's efficacy in detecting fraudulent transactions.

Before evaluating the model, it was necessary to set a threshold for anomaly detection. The threshold was set at the 95th percentile of the mean squared error (MSE) loss for the test dataset. This threshold was selected because it is a commonly used value for anomaly detection in credit card transaction datasets, and it lowered the risk and cost of misclassifying a fraudulent transaction into a valid transaction.

After setting the threshold, the anomalies were filtered out of the dataset. The model was then evaluated using the confusion matrix, precision-recall curve, and ROC curve. The confusion matrix shows the number of true positives, false positives, true negatives, and false negatives for the model. A high proportion of true positives compared to false negatives was observed, indicating that the model is effective at detecting fraudulent transactions. Although there is a high number of false positives, misclassifying some valid transaction records as fraudulent is acceptable since the cost is not as high as misclassifying a truly fraudulent record as a valid record in anomaly detection scenarios.

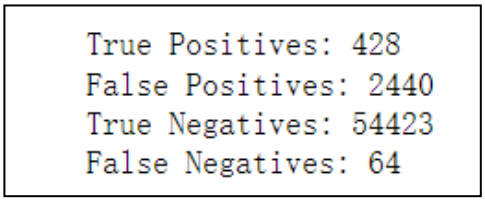


Fig. 7. Confusion Matrix

The precision-recall curve shows the trade-off between precision and recall for different thresholds. The high recall will lead to poor precision, and vice versa. In credit card fraud detection, it is desirable to have high recall at the expense of precision since missing a fraudulent transaction could have severe consequences. The Precision-Recall Curve is lying on the diagonal from upper left to bottom left indicating a reasonable balance of the two schemes for the model.

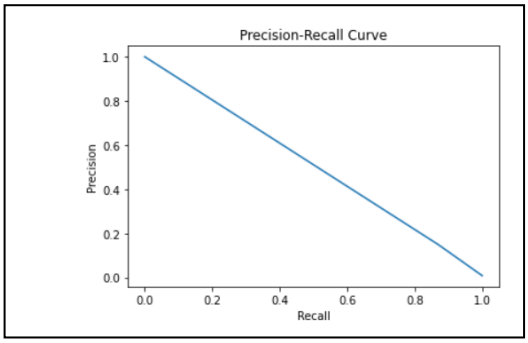


Fig. 8. Precision-Recall Curve

The ROC curve plots the true positive rate against the false positive rate for different classification thresholds. The area under the curve (AUC-ROC) is a commonly used metric to evaluate classification models, with a score of 0.5 indicating random guessing and a score of 1 indicating perfect classification. The AUC-ROC score for the autoencoder model was 0.91, indicating good performance. The advantages of using all three metrics are that they provide a comprehensive view of the model's performance, allowing for a more informed decision about its efficacy in detecting fraudulent transactions.

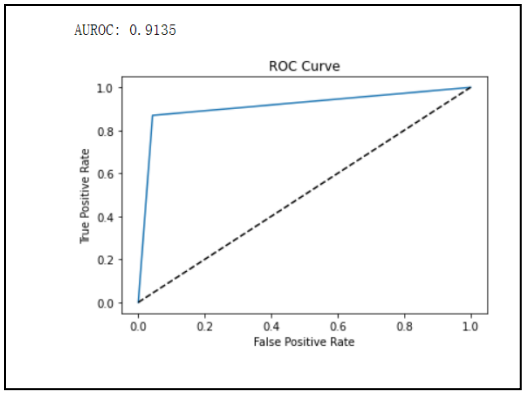


Fig. 9. AUC-ROC score and ROC Curve

IV. Conclusion

In conclusion, we have developed an autoencoder-based anomaly detection model for credit card fraud detection. We have implemented hyperparameter tuning with Optuna to find the best parameter combination, which improved the model's performance in reconstructing valid transactions. We then trained the model with the best parameter combination and tested its performance using a scatter plot of the mean squared error (MSE) loss against the record ID. We found that most of the data had an MSE loss lower than 50, while some had a higher loss, indicating that those were likely to be fraudulent transactions.

To evaluate the model performance, we set the threshold of the MSE loss to the 95th percentile and used three metrics: a confusion matrix, a precision-recall curve, and a ROC curve. We found that the model had a high proportion of true positives compared to false negatives, which is important in fraud detection since the cost of misclassifying fraudulent transactions as valid ones are high. The precision-recall curve showed a balance between precision and recall, while the ROC curve demonstrated the model's excellent performance, with an AUC-ROC score of 0.91.

Overall, our model has shown promising results in detecting credit card fraud. With further optimization and more extensive datasets, we believe that our model can be further improved and applied to other financial anomaly detection tasks. There are several future research directions that can be explored to further improve the performance of credit card fraud detection models. One promising direction is to incorporate graph neural networks (GNNs) to model the relationship between transactions and detect more complex fraud patterns, such as collusion fraud. Another direction is to explore more advanced techniques, such as adversarial training, to improve the model's robustness against adversarial attacks.

References

[1] Nilson Report: News and statistics for card and mobile payment executives. Nilson Report | News and Statistics for Card and Mobile Payment Executives. (n.d.). Retrieved April 7, 2023, from https://nilsonreport.com/

[2] ULB, M. L. G.-. (2018, March 23). Credit Card Fraud Detection. Kaggle. Retrieved April 7, 2023, from https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

[3] Winastwan, R. (2021, December 14). Hyperparameter tuning of neural networks with Optuna and pytorch. Medium. Retrieved April 7, 2023, from https://towardsdatascience.com/hyperparameter-tuning-of-neural-networks-with-optuna-and-pytorch-22e179efc837

APPENDIX

Source Code Link:
https://github.com/LKUGB/Final_Deliverable_Applied_AI_Jin