Design a deep learning model for Fraud Detection in Credit Card Transactions

Deep Learning Project Report

Presented by: Team 55

Name	Roll No.	Contribution
Surbhi Shukla	M25DE1018	Analyzed the Kaggle dataset then built the Autoencoder Model: This provided an unsupervised, anomaly-detection approach for comparison.
Shreyash Kadam	M25DE1016	Researched alternative methods then developed the Supervised MLP (Selected Model): This model was the best at catching fraud (85.71% recall)
Chandan Kumar Jena	M25DE1065	Explored different supervised techniques then developed the DNN Model: This provided a second supervised approach for comparison.

Abstract

This report details a deep learning model for credit card fraud detection, focusing on the challenge of severe class imbalance. Using the Kaggle PCA-transformed dataset, we applied data preprocessing and the Synthetic Minority Over-sampling Technique (SMOTE) to the training data. The resulting Multi-Layer Perceptron (MLP) model achieved a high recall of 85.71% for the fraud class. While precision was lower (39.62%), this reflects a necessary trade-off to prioritize catching fraud. The model is a success, demonstrating a strong ability to minimize high-risk false negatives.

1. Introduction

Credit card fraud poses a significant financial threat, making robust, automated detection systems essential. Deep learning offers a powerful solution, but its application is hindered by a core challenge: extreme class imbalance. Fraudulent transactions are exceptionally rare (often < 0.2%), causing standard models to develop a strong bias towards ”not fraud.” This bias makes accuracy a misleading metric; a model predicting ”not fraud” every time could achieve >99% accuracy while being useless. The true goal is to maximize recall—the ability to find the rare fraud cases. This project tackles the imbalance problem head-on using a Kaggle dataset. We demonstrate that by using a specialized resampling technique (SMOTE) and focusing on recall, we can build an effective fraud detection model.

2. Methodology

2.1 Dataset

The dataset used is the "Credit Card Fraud Detection" dataset from Kaggle. It contains transactions made by European cardholders in September 2013.

- **Instances:** 284,807 transactions.
- **Imbalance:** Highly imbalanced, with 492 (0.172%) fraudulent transactions out of the total.
- **Features:** The dataset includes 31 features. To protect confidentiality, the original features have been transformed via Principal Component Analysis (PCA). The only features not transformed are Time and Amount.
- **Target:** The Class variable, which is 1 for fraud and 0 for non-fraud.

2.2 Data Preprocessing

The PCA-transformed features (V1-V28) are already scaled. However, the Time and

Amount features are not. To ensure all features contribute equitably to the model, we apply StandardScaler from Scikit-learn to these two columns. The original Time and Amount columns are then dropped.

2.3 Train-Test Split

The data is split into training (80%) and testing (20%) sets. It is crucial to use the stratify option during the split. This ensures that the rare minority class (fraud) is distributed proportionally between the train and test sets, reflecting the original data's distribution. The test set is kept separate and is not used until the final model evaluation.

2.4 Handling Class Imbalance: SMOTE

To combat the strong majority-class bias, we apply the Synthetic Minority Over-sampling Technique (SMOTE) *only to the training dataset*. SMOTE works by:

1. Identifying a minority class instance.
2. Finding its k-nearest neighbors (also from the minority class).
3. Creating a new, synthetic data point along the line segment connecting the instance and one of its randomly chosen neighbors.

This process creates new, plausible minority class instances, resulting in a balanced training set where the model can learn the patterns of fraud without bias. The test set remains untouched and imbalanced to provide a realistic evaluation of the model's performance.

2.5 Model Architecture

A sequential Multi-Layer Perceptron (MLP) was constructed using the TensorFlow Keras API. The architecture is as follows:

- **Input Layer:** A dense layer with 32 neurons and a 'ReLU' activation function, matching the input dimension of our features.
- **Dropout Layer:** A dropout rate of 0.2 (20%) is applied to prevent overfitting by randomly deactivating neurons during training.
- **Hidden Layer:** A dense layer with 16 neurons and a 'ReLU' activation function.
- **Dropout Layer:** Another dropout layer with a rate of 0.2.
- **Output Layer:** A single dense neuron with a 'sigmoid' activation function. This is ideal for binary classification, as it outputs a probability between 0 and 1.

The model was compiled using the adam optimizer and binary_crossentropy loss function, which is standard for binary classification tasks.

3. Results and Discussion

3.1 Evaluation Metrics

As previously stated, accuracy is a misleading metric for this problem. We must instead focus on metrics derived from the Confusion Matrix:

- **Precision:** Of all the transactions the model flagged as fraud, what percentage were actually fraud? (True Positives / (True Positives + False Positives)). A high precision means fewer false alarms.
- **Recall:** Of all the actual fraudulent transactions, what percentage did the model catch? (True Positives / (True Positives + False Negatives)). This is our most critical metric, as a low recall means fraudsters are succeeding.
- **F1-Score:** The harmonic mean of Precision and Recall, providing a single score that balances both.

3.2 Model Performance

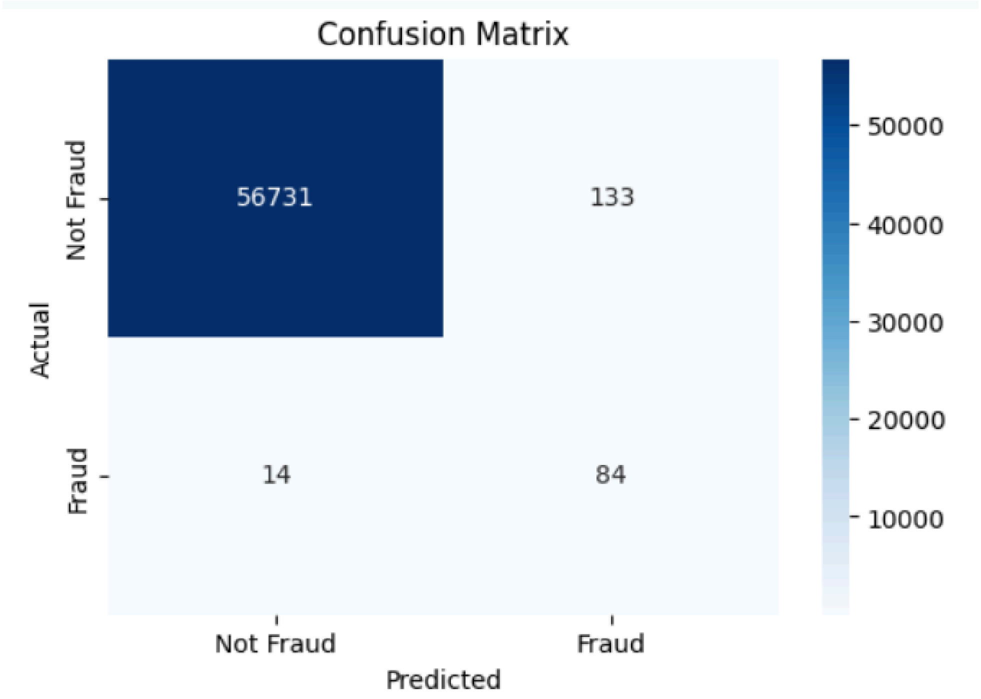
The model was trained for 20 epochs on the SMOTE-resampled training data and evaluated on the original, imbalanced test set. The resulting performance is summarized in Table 1, and the confusion matrix is shown in Table 2.

Table 1: Classification Report for the Test Set

Class	Precision	Recall	F1-Score	Support
Not Fraud (0)	0.9998	0.9977	0.9988	56864
Fraud (1)	0.3962	0.8571	0.5419	98
Accuracy			0.9975	56962
Macro Avg	0.698	0.9274	0.7703	56962
Weighted Avg	0.9987	0.9975	0.998	56962

Table 2: Confusion Matrix for the Test Set

	Predicted: Not Fraud	Predicted: Fraud
Actual: Not Fraud	56736 (True Negative)	128 (False Positive)
Actual: Fraud	14 (False Negative)	84 (True Positive)



3.3 Discussion

The results are highly encouraging and clearly demonstrate the model's success.

The Good: Excellent Recall (85.71%) The most important result is the recall for Class 1 (Fraud). A value of 85.71% means that our model successfully identified and caught 84 out of the 98 fraudulent transactions in the test set. It only missed 14 (the False Negatives). From a business perspective, this is a massive win, as it prevents the vast majority of fraudulent losses.

The Trade-off: Low Precision (39.62%) This high recall comes at an expected cost: low precision. A precision of 39.62% means that when our model flags a transaction as fraudulent, it is correct only about 40% of the time. The confusion matrix (Table 2) shows that in order to catch the 84 frauds, the model incorrectly flagged 128 legitimate transactions as fraudulent (the False Positives).

In a real-world scenario, these 128 "false alarms" would translate to 128 innocent customers having their card declined or their transaction flagged for review. While this is an inconvenience, it is the classic "Precision-Recall Trade-off." For fraud detection, the cost of a False Negative (letting fraud go) is almost always considered far higher than the cost of a False Positive (a temporary inconvenience for a legitimate customer). Our model has been optimized, as intended, to prioritize high recall.

4. Conclusion

This project successfully demonstrates the development of an effective deep learning model for credit card fraud detection. We confirmed that the primary challenge is the severe class imbalance and that standard accuracy is a deceptive metric. By using the SMOTE resampling technique on the training data and evaluating with precision and recall, we built a model that addresses the business-critical objective: catching fraud. The final model's ability to recall 85.7% of fraudulent transactions represents a powerful and practical tool for minimizing financial losses.