CREDIT CARD FRAUD DETECTION

A PREPRINT

Hasan Evci
Department of Computer Science
University of Stuttgart
Stuttgart, PA 70511
st160883@stud.uni-stuttgart.de

Tareq Abu El Komboz
Department of Computer Science
University of Stuttgart
Stuttgart, PA 70569
st161420@stud.uni-stuttgart.de

February 8, 2024

ABSTRACT

Improving algorithms for credit card fraud detection can save institutions and private persons tremendous amounts of money and is of huge importance. In this work, we show that deep learning methods are superior to many traditional anomaly detection methods in the field of credit card fraud detection. Using the Credit Card Fraud Detection Dataset 2023 with over 550,000 credit card transactions, we found that nearly all proposed deep learning architectures achieved better metrics than traditional baselines, suggesting their greater efficacy for high-dimensional data. The best-performing model overall is the one-class neural network with a macro-average F1-score of 92%, whereas the best-performing baseline model is the Mahalanobis distance method with a macro-average F1-score of 91.7%. Interestingly, distance-based methods such as the kNN or Mahalanobis distance methods slightly outperform advanced variations of the autoencoder, indicating that simpler models can still be highly effective. This finding emphasizes the importance of considering both advanced and traditional approaches in fraud detection strategies.

Keywords: one-class neural networks, autoencoder, anomaly detection, credit card fraud, deep learning, unsupervised learning.

1 Introduction

In the rapidly evolving digital field of finance, the increase in card transactions has been paralleled by a corresponding rise in fraudulent activities. Proactive forecasting and prevention of such frauds are essential in mitigating losses, a point underscored by a recent report[1]: issuers, merchants, and acquirers suffered losses of $32.34 billion due to credit card fraud. Notably, the United States, despite accounting for only 23.02% of global transaction volume, bore a disproportionate 36.83% of these losses, highlighting the urgency and complexity of the issue.

The traditional methods, often labor-intensive and time-consuming, are proving inadequate in the face of sophisticated fraud techniques. This is where the potential of deep learning comes into play. By automating the fraud detection process, deep learning can cut operational costs, enhance accuracy, and accelerate the detection process.

There are several previous works in the active research area of credit card fraud detection [1, 2, 3]. Especially autoencoders [4, 5, 6] are increasingly recognized as effective tools in this domain for their ability to detect fraudulent transactions through learned representations of data.

In this study, we tackle the challenge of detecting unusual patterns in credit card transactions using the Credit Card Fraud Dataset 2023. Our focus is on utilizing deep learning methods to identify fraudulent transactions. Autoencoders and one-class neural networks are effective in detecting transactions that don't follow the typical patterns, making them ideal for this task. Additionally, we compare their performance with well-known methods like Isolation Forest and k-Nearest Neighbors (kNN) distance. These comparisons help us assess the effectiveness of deep learning methods against traditional anomaly detection techniques, offering a concise yet comprehensive view of their potential to enhance

[1] https://nilsonreport.com/newsletters/1232/

credit card fraud detection. Thus, the research question this work aims to investigate and answer is the following: *How do deep learning methods compare to traditional anomaly detection methods in effectively identifying credit card fraud?*

2 Anomaly detection

An anomaly is an observation that is significantly different from what is considered normal. The distinction between noise and anomalies is not clear-cut and often depends on the analyst's interest. In general, anomaly detection methods provide two types of results: *outlier scores* and *binary labels* [7]. Outlier scores rank data points by how much they differ from the norm, while binary labels simply mark data as normal or an outlier. In the following, we provide a formal definition of anomaly detection and point out challenges in anomaly detection.

2.1 Formal definition

To formally define anomaly detection, many researchers in the past have used probability theory. We use the definition provided by Ruff et al. [4]: Let $\mathcal{X} \subseteq \mathbb{R}^D$ be the data space of the task at hand.

Normality refers to the typical behavior in a specific situation or application, represented as a distribution called $\mathbb{P}^+$. $\mathbb{P}^+$ also has a corresponding probability density function (pdf) $p^+(x)$.

An *anomaly* is a data point (or a group of points) that falls into a region where the probability of occurrence under $\mathbb{P}^+$ is very low. We can define this set of anomalies as

$$\mathcal{A} = \{x \in \mathcal{X} \mid p^+(x) \leq \tau\}, \quad \tau \geq 0,$$

where $\mathcal{A}$ consists of data points $x \in \mathcal{X}$ that satisfy the condition $p^+(x) \leq \tau$, with τ being a threshold value. It is set such that the probability of $\mathcal{A}$ happening under $\mathbb{P}^+$ is considered "sufficiently small".

2.2 Challenges

The quest to effectively distinguish between *normal* and *anomalous* events in anomaly detection is fraught with challenges, making it a complex task. A primary difficulty arises from the substantial variability innate in normal data. This variation can be so extensive that it leads to the misclassification of normal instances as anomalies (type I error) or, conversely, fails to identify actual anomalies (type II error) [4]. Further, features with wide value ranges, noise, or irrelevance can skew distance computations and obscure the presence of anomalies [4].

Additionally, the inherent rarity of anomalies in data leads to imbalanced datasets and often results in unlabeled data, making it unclear which points are truly anomalous. This situation often necessitates the use of unsupervised learning to build models based on the majority of data points. Moreover, the diverse settings in anomaly detection, each with its unique challenges, add to the complexity of the task. For a detailed discussion on the different settings of anomaly detection, refer to the Appendix A.1.

2.3 Methods for anomaly detection

Anomaly detection approaches are divided into four main categories: classification, probabilistic, reconstruction, and distance-based methods. Classification methods, such as one-class support vector machines, learn a region (or boundary) of normal data points and flag data points as anomalies when they fall outside this region. Probabilistic techniques, such as Gaussian mixture models, identify anomalies by deviations from statistical distributions. Reconstruction methods, exemplified by autoencoders, flag anomalies based on the inability to accurately reconstruct data. Distance-based approaches, including k-nearest neighbors, detect outliers by measuring distances in data space. These categories are split further into shallow (classical machine learning techniques) and deep methods (neural networks and deep learning). Deep methods are particularly adept at handling complex, high-dimensional data. For more detailed information on these approaches, Appendix A.2 offers an extensive overview.

3 Methods

In this section, we detail the comprehensive choices and strategies implemented in our study, providing all the necessary information to ensure the scientific integrity and reproducibility of our work[2].

[2]The implementation and additional details can be found here: `https://github.com/Hasosh/Credit-Card-Fraud-Detection`

3.1 Data Preprocessing

We have excluded the 'id' feature from the dataset, as it serves merely as a unique identifier for each data sample. Furthermore, we found out that all features are already normalized, except for the "amount" feature. We standardize the "amount" feature using Z-score normalization. We also implemented and tested additional scaling methods, specifically MinMax and Robust scaling. However, we have not continued using them as they have not led to a performance increase across the models.

In our data preprocessing, we deliberately choose not to employ dimensionality reduction techniques. The rationale behind this decision is rooted in our belief that more information typically leads to better results. Even features contributing marginally to the final classification hold value and can enhance the model's performance. Additionally, this strategy allows us to explore the performance of shallow versus deep models in handling high-dimensional data.

Instead, we also explored a hybrid approach. This involved encoding features using our optimally performing autoencoder and then feeding these encoded features into other anomaly detection methods, such as the OC-SVM. This strategy aims to leverage the autoencoder's ability to capture meaningful data representations, potentially boosting the effectiveness of subsequent anomaly detection processes.

3.2 Approach

In our study, we begin with the evaluation of baseline models to establish a benchmark for feasible results in anomaly detection. We select a diverse range of baselines, including K-Means clustering, One-Class SVM (OC-SVM), kNN distance, Isolation Forest, Mahalanobis Distance, Gaussian Mixture Model (GMM), and Kernel Density Estimator (KDE). This selection ensures a comprehensive comparison of methods from the aforementioned four categories (see Section 2.3). For more detailed information about these models, please refer to the Appendix A.3.

Subsequently, we employ deep learning models, particularly autoencoders and OC-NNs, known for their effectiveness in anomaly detection. Autoencoders are used to capture the normal data distribution by learning to reconstruct input data. Anomalies are then identified based on the reconstruction error; the larger the error, the more likely the data point is an anomaly. OC-NNs [8], on the other hand, are very similar to OC-SVMs. The key difference is that they learn a decision boundary in a neural network setting, which separates normal data from anomalies. The decision is based on whether new data falls within this learned region. More information about the deep learning models can be found in the Appendix A.4.

Next to the standard autoencoder, we use two more sophisticated variants: The denoising and the variational autoencoder. Our autoencoders use a two-layer encoder and decoder, with dimensions going from 28 to 16 to 8, and then inversely for the decoder. Each layer includes batch normalization (for regularization) and ELU activation. To compute the reconstruction loss, we use the standard mean squared error loss:

$$\ell(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^{N} \|x_i - \hat{x}_i\|^2$$

There are different strategies to set the threshold of the autoencoders, e.g. fixed, statistical, percentile-based etc. For our dataset, we found out that setting the threshold to the 90%-percentile of the training reconstruction loss distribution yields the best results.

For the OC-NN, our design incorporates a single 12-unit hidden layer and a sigmoid-activated output layer, which is dimensionally set to 1. It leverages a unique loss function akin to that of the OC-SVM. The overall optimization objective is the following [8]:

$$\min_{w,V,r} \frac{1}{2}\|w\|_2^2 + \frac{1}{2}\|V\|_F^2 + \frac{1}{\nu} \cdot \frac{1}{N} \sum_{n=1}^{N} max(0, r - \langle w, g(VX_{n:})\rangle) - r$$

The optimization objective of the OC-NN closely mirrors that of the OC-SVM, with a key distinction lying in the method of kernel approximation. While the OC-SVM relies on a predefined kernel for mapping data into a higher-dimensional space, the OC-NN seeks to learn this mapping dynamically through a neural network. This approach allows the OC-NN to potentially adapt better to complex data structures, as it is not constrained by the limitations of a fixed kernel choice. Also, note that the optimization objective is non-convex. Thus, it is not guaranteed that we find the global optimum. The optimization is done in 2 steps: First fixing r and then optimizing for w and V, then solving r using the ν-quantile of the training scores $\langle y_n \rangle$. Knowing r, we can compute OC-NN's decision function, which classifies data points with positive anomaly scores as normal, with negative anomaly scores as fraudulent:

$$S_n = sgn(\hat{y}_n - r)$$

All deep learning models are trained using the Adam optimizer. To optimize these deep learning architectures, we perform a parameter search. The different hyperparameters we tried are summarized in Appendix D. Additionally, we implement early stopping and batch normalization as regularization techniques during the training of these models. This approach helps to prevent overfitting, ensuring that our models generalize well to unseen data, thereby enhancing their reliability and effectiveness in detecting anomalies in credit card transactions. Further implementation details are described in Appendix C.

4 Experiments and Results

This section presents an overview of the dataset used, our overall evaluation procedure, and the results we obtained from our analyses.

4.1 Dataset

The Credit Card Fraud Detection Dataset[3] 2023 features over 550,000 European credit card transactions, aimed at binary classification to identify fraudulent activities. It includes a unique identifier and 28 anonymized features, which represent various transaction attributes like time and location, along with the transaction amount for each transaction. For computational reasons, our study uses a modified version of this dataset with fewer samples.

A key challenge in using this dataset is its high dimensionality, requiring computational resources and potentially advanced feature selection. Furthermore, the anonymization of features complicates data interpretability as the underlying patterns and relationships among these attributes are obscured, making it difficult to apply domain-specific knowledge or perform intuitive analysis. Lastly, its semi-supervised nature demands models capable of learning from normal data instances.

4.2 Evaluation

Various metrics are used in anomaly detection literature [4, 7, 3]. In this work, we define anomalies to be the positive class. We have selected traditional metrics such as precision, recall, and F1-score for evaluation. In addition, we employ the Matthews Correlation Coefficient (MCC), a robust measure for binary classification quality assessment. The Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) plot is also used to evaluate model performance across different thresholds. Details on these metrics are available in the appendix B. Furthermore, to ensure the robustness and reproducibility of our results, each baseline model is run 10 times and each deep learning model is trained using 10-fold cross-validation. Here, we report macro-averaged results.

4.3 Results

In this subsection, we report the results we achieved while comparing our DL models against the selected baseline models. Our results are compactly presented in Table 1. Every row corresponds to a different model and every column corresponds to one metric. For each metric, we highlighted the model that performed best in bold font. The best-performing model is the DL model OC-NN, whereas the worst-performing is the Isolation forest model. The best baseline model is the Mahalanobis distance model.

To refine the understanding of feature influence within our top-performing model, the OC-NN, we undertook an analysis using Shapley values. We found out that there are features that have notably more influence on the model's prediction than others. The most impactful features and the least impactful features are summarized in Figure 4.3.

The results of our hybrid approach are summarized in Figure 2. Upon examination, it becomes apparent that there was a significant decline in performance metrics across all models when utilizing the hybrid method. Additional results and plots are all provided in Appendix E.

[3] `https://www.kaggle.com/datasets/nelgiriyewithana/credit-card-fraud-detection-dataset-2023`

[4] To calculate the ROC-AUC curve, we need the algorithm to output anomaly scores. K-means does not have anomaly scores like the other models and therefore its entry for the ROC-AUC field is left empty.

[5] SGD OC-SVM is a variant of the OC-SVM that employs Stochastic Gradient Descent (SGD) instead of quadratic programming for optimization.

	Model	Precision	Recall	F1-Score	ROC-AUC	MCC
Baselines	Naive	0.5	1.0	0.66	0.5	0
	K-Means (k=2)[4]	**1.0**	0.764	0.866	–	0.786
	OC-SVM	0.653	0.937	0.77	0.89	0.49
	SGD OC-SVM[5]	0.682	**0.981**	0.805	**0.973**	0.582
	Isolation Forest	0.946	0.64	0.763	0.917	0.638
	kNN distance	0.937	0.842	0.887	0.949	0.79
	Mahalanobis Distance	0.945	0.89	0.917	0.947	0.84
	GMM	0.958	0.859	0.905	0.958	0.825
	KDE	0.682	0.978	0.803	0.941	0.579
DL Models	Autoencoder	0.899	0.895	0.896	0.941	0.793
	DAE	0.896	0.882	0.889	0.93	0.78
	VAE	0.891	0.828	0.859	0.918	0.729
	OC-NN	0.974	0.871	**0.92**	0.929	**0.853**

Table 1: Performance of baseline models and DL Models

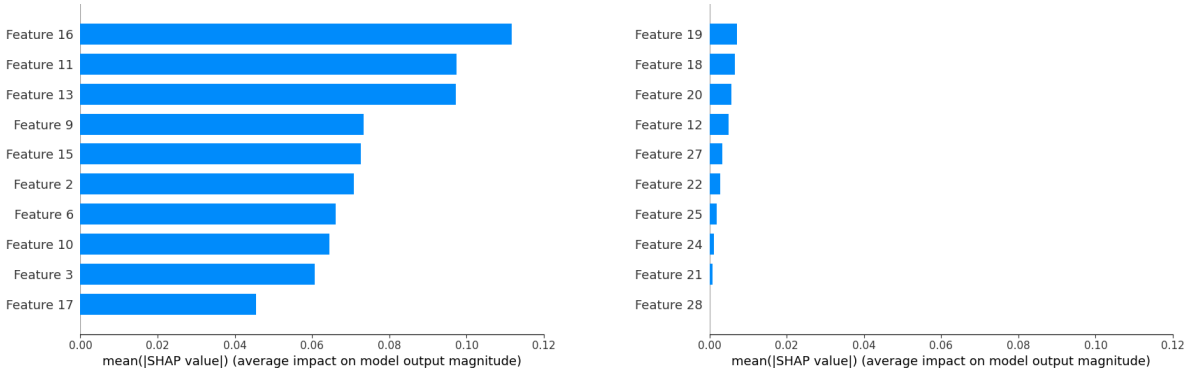

Figure 1: Most impactful features (left) and least impactful features (right) of the OC-NN model.

5 Discussion

In this section, we delve into a critical examination of our findings, exploring potential explanations for our results and identifying opportunities for further refinement and expansion of our research in the field of anomaly detection and credit card fraud detection.

5.1 Interpretation of the Results

The OC-NN emerged as the top performer among all models, likely due to its unique design tailored for anomaly detection. Unlike other models, the OC-NN specifically shapes its data representation within the hidden layer to effectively identify anomalies [8]. This approach provides a unique advantage in distinguishing between normal and anomalous patterns in complex datasets.

Autoencoder-based models in the study showed consistent performance, with advanced versions like DAE and VAE not outperforming simpler autoencoders. This suggests that the dataset likely had low noise levels, as DAEs are designed for noisy data and might reconstruct anomalies too well, reducing their effectiveness. VAEs, known for their generalization abilities, didn't show improved performance either. This indicates that the test data closely resembles the training data, rendering their stochastic nature less beneficial in our context.

The superior performance of deep learning models over traditional baselines across almost all baseline models is likely attributed to deep learning's resilience against the curse of dimensionality, a common challenge in high-dimensional data spaces like credit card transactions. Deep learning architectures, with their layered, complex structures, are better at understanding and making sense of complex data, leading to more precise and reliable detection of anomalies.

Surprisingly, baseline models such as the kNN distance, Mahalanobis distance, and GMMs marginally surpassed more complex autoencoder variants, demonstrating that less complex models remain quite effective in certain scenarios. This

outcome highlights the significance of incorporating a blend of both deep and shallow methods in devising strategies for fraud detection.

Contrary to our initial assumptions, the most intriguing insight was the minimal influence of feature 28—representing the transaction amount and notably the sole unmasked feature—on the model's predictive capabilities. This finding defies the conventional expectation that higher transaction amounts might be indicative of fraudulence.

In the hybrid method, performance significantly decreased. The autoencoder appears to have developed a feature space overly specialized for its reconstruction process, which reduced the interpretability for other models, notably the OC-NN. The OC-SVM and kNN methods still performed well, showing their resilience to changes in feature representation, which is consistent with OC-SVM's frequent use in hybrid models.

5.2 Future Work

For future work, we aim to investigate the underlying reasons for the performance decline observed with the hybrid method, with a particular focus on understanding how the autoencoder's specialized feature space impacts the interpretability and effectiveness of various models. Furthermore, we focused on the semi-supervised setup with only normal data for training, assuming the absence of anomalies in the training set for all models. It would be interesting to examine if other setups with fraudulent data in the training data set allow for better performances. Additionally, exploring the application of ensemble techniques presents a promising avenue for potentially making predictions more robust.

6 Conclusion

In conclusion, this study in credit card fraud detection reveals that deep learning models generally outperform traditional methods. This suggests their greater efficacy, especially in dealing with the challenges posed by high-dimensional data. The One Class Neural Network emerges as the most effective model, with significant proficiency in identifying fraudulent transactions. While no model reaches absolute precision or recall, the findings underscore the inherent challenge of distinguishing anomalous from normal transactions due to overlapping data distributions. Interestingly, traditional models like kNN distance and GMMs show specific strengths in certain metrics, suggesting their continued relevance. The similar performance of various autoencoder models suggests that data noise levels are low and that sophisticated models like DAEs and VAEs do not necessarily offer additional benefits in this context.

References

[1] Asma Cherif, Arwa Badhib, Heyfa Ammar, Suhair Alshehri, Manal Kalkatawi, and Abdessamad Imine. Credit card fraud detection in the era of disruptive technologies: A systematic review. *J. King Saud Univ. Comput. Inf. Sci.*, 35(1):145–174, 2023.

[2] Vaishnavi Nath Dornadula and Sa Geetha. Credit card fraud detection using machine learning algorithms. *Procedia computer science*, 165:631–641, 2019.

[3] C Victoria Priscilla and D Padma Prabha. Credit card fraud detection: A systematic review. In *Intelligent Computing Paradigm and Cutting-edge Technologies: Proceedings of the First International Conference on Innovative Computing and Cutting-edge Technologies (ICICCT 2019), Istanbul, Turkey, October 30-31, 2019 1*, pages 290–303. Springer, 2020.

[4] Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proc. IEEE*, 109(5):756–795, 2021.

[5] Sumit Misra, Soumyadeep Thakur, Manosij Ghosh, and Sanjoy Kumar Saha. An autoencoder based model for detecting fraudulent credit card transaction. *Procedia Computer Science*, 167:254–262, 2020.

[6] Hosein Fanai and Hossein Abbasimehr. A novel combined approach based on deep autoencoder and deep classifiers for credit card fraud detection. *Expert Systems with Applications*, 217:119562, 2023.

[7] Charu C. Aggarwal. *An Introduction to Outlier Analysis*, pages 1–34. Springer International Publishing, Cham, 2017.

[8] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *CoRR*, abs/1802.06360, 2018.

[9] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4), 2016.

A Additional information on anomaly detection

This section provides further information on anomaly detection.

A.1 Settings

Anomaly detection comes with different settings [9], each with its own set of challenges and strategies. In the following, we summarize them.

- *Supervised Anomaly Detection*: This setup utilizes labeled data to train a classifier for identifying anomalies, similar to traditional pattern recognition but challenged by class imbalance. It isn't very common in practice because anomalies are often not known or labeled in advance and may appear unexpectedly during testing.
- *Semi-supervised Anomaly Detection*: Here, only normal data is used for training, aiming to detect anomalies as deviations from this learned norm, often employing methods like One-class SVMs and autoencoders.
- *Unsupervised Anomaly Detection*: This setup requires no labels. It solely relies on the data's intrinsic characteristics to discern anomalies, typically using distance or density measures.

Each setup has its advantages, complexities, and areas of application, depending on the nature of the data and the specific requirements of the task at hand. We focus on the semi-supervised setup in the scope of this project.

A.2 Approaches

Models for anomaly detection can be categorized into four different types as can be seen in Figure 2.

- *Classification Models*: These models learn a decision boundary that includes most normal data points. All points outside of this learned boundary are classified as anomalies. A typical classification model is the one-class SVM.
- *Probabilistic Models*: These models calculate the probability of a data point belonging to a certain class or distribution. If the probability is extremely low, the data point is considered an anomaly. Examples include the Mahalanobis Distance, Kernel Density Estimators, and Gaussian Mixture Models.
- *Reconstruction Models*: These are typically unsupervised learning models that learn to reconstruct normal data points. During testing, if a data point cannot be accurately reconstructed, it is considered an anomaly. The k-means algorithm and Autoencoders are common examples of this type of model.
- *Distance-Based Models*: These models identify anomalies based on the distance or similarity between data points. If a data point is significantly distant or dissimilar to most of the other data points, it is considered an anomaly. Examples include kNN distance and Isolation Forests.

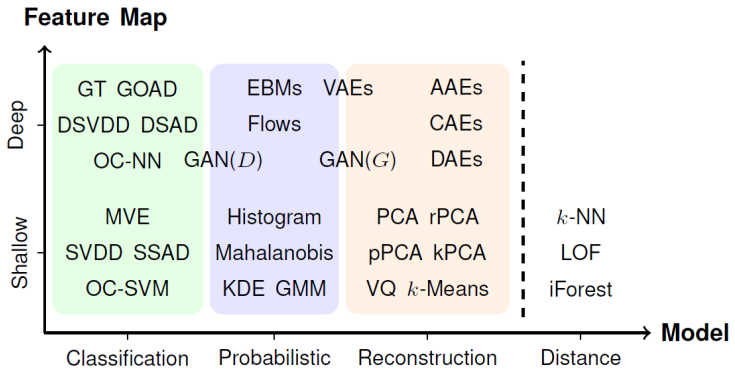

Figure 2: Anomaly detection approaches [4].

A.3 Shallow models

Here we briefly explain the different baseline models and algorithms we used for our comparisons.

K-Means Clustering is a widely used method for partitioning data into distinct groups or clusters based on feature similarity. The algorithm operates by iteratively assigning each data point to the cluster whose centroid is nearest, and then recalculating the centroids of these clusters. In the context of anomaly detection, K-Means is particularly effective because anomalies are typically identified as points that have a substantial distance from the centroid of their assigned cluster, suggesting a deviation from the common patterns within that cluster.

One-Class Support Vector Machine (OC-SVM) is an unsupervised learning algorithm primarily used for anomaly detection. OC-SVM works by learning a decision function that delineates the normal data points, effectively isolating them from the rest. The key to OC-SVM is its ability to create a decision boundary that maximizes the margin between the data points and the origin, thus separating most of the data points from the origin in the feature space. Data points on the other side of this boundary are classified as anomalies.

The k-Nearest Neighbor (kNN) Distance method is a straightforward yet effective approach for identifying outliers. It relies on the distance of a data point from its nearest neighbors to assess its anomalous nature. The underlying assumption is that normal points have a proximity to their neighbors, whereas outliers are typically further away. For each data point, the distance to its k nearest neighbors is calculated and analyzed. Anomalies are those points that exhibit a significantly larger average distance to these neighbors compared to the rest of the data points.

The *Isolation Forest* algorithm offers a unique perspective on anomaly detection. It operates on the principle that anomalies are "few and different", making them easier to isolate compared to normal points. This method involves randomly selecting a feature and a split value between the feature's maximum and minimum values, and then partitioning the dataset accordingly. The anomalous points are those that are isolated with fewer partitions, suggesting their rarity and distinctness in the dataset.

The *Mahalanobis distance* is a measure of the distance between a point and a distribution. It's a multivariate generalization of the square of the standard score.

A *Gaussian Mixture Model (GMM)* is a probabilistic model that assumes that the data points in a dataset come from a mixture of multiple Gaussian distributions. Each Gaussian distribution represents a cluster or component in the data, and the parameters of the distributions are estimated from the data.

Kernel Density Estimators (KDEs) estimate the probability density function of the data points in a sample space. If we consider that the norm of a dataset should fit a certain kind of probability distribution, the anomalies are those that we should see rarely, or in a very low probability.

A.4 Deep learning models

We focus on utilizing and explaining deep learning methods for the problem of anomaly detection in this study.

Autoencoders are a type of neural network used for unsupervised learning, focusing on learning efficient representations of unlabeled data. They consist of two main parts: an encoder, which compresses high-dimensional input into a low-dimensional latent space, and a decoder, which reconstructs the input from this compressed form. The goal is to minimize the difference between the original input and its reconstruction, with the training primarily on normal data. This leads to higher reconstruction errors for anomalies, as they deviate from the learned norm, making autoencoders effective for anomaly detection by using this error as an anomaly score. Thus, a data point is classified as an anomaly if its reconstruction error is above a certain threshold. The general architecture of an autoencoder can be seen in Figure A.4.

Denoising Autoencoders (DAEs) are a variant where the input data is intentionally corrupted with noise before being fed into the network. The autoencoder then learns to reconstruct the original, uncorrupted data from the noisy input, enhancing its ability to identify and disregard irrelevant variations in the data.

Variational Autoencoders (VAEs) are a type of autoencoder that learns to map input data to a probabilistic distribution in the latent space. Unlike standard autoencoders, VAEs are stochastic, meaning they generate different encodings for the same input by sampling from this distribution. This feature makes them particularly effective for anomaly detection, as they identify anomalies by detecting deviations from the learned probabilistic distribution of normal data. The general architecture of a VAE is depicted in Figure A.4.

One-class neural networks (OC-NNs) are utilized for anomaly detection and operate similarly to the one-class SVM. The OC-NN's loss function mirrors that of the one-class SVM, focusing on creating a separation between normal data points and the origin in a given feature space. This loss function balances the model's complexity with its ability to distinguish between normal and anomalous data points. In essence, OC-NN adapts the one-class objective of SVMs

[6]Lilian Weng. From autoencoder to beta-vae. lilianweng.github.io, 2018.

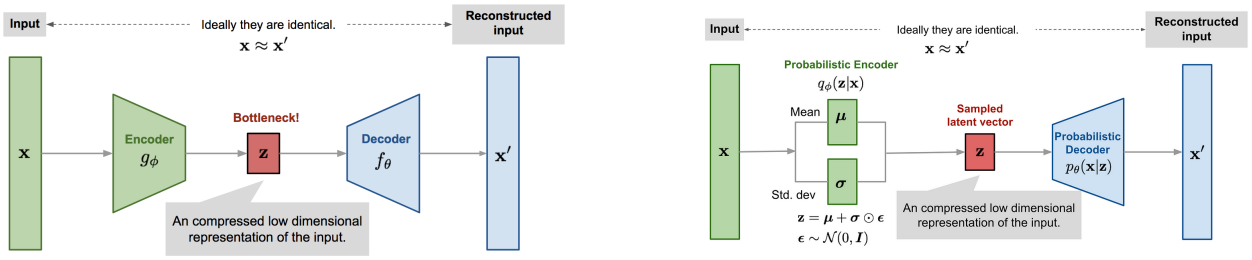

Figure 3: Standard autoencoder architecture (left) and variational autoencoder architecture (right) [6].

into a neural network architecture, leveraging the neural network's ability to learn complex data representations for effective anomaly detection.

B Metrics

In this part of the appendix, we give the definitions of the metrics we used to evaluate our experiments.

Precision, or Positive Predictive Value, reflects the accuracy of positive predictions.

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

True Positive Rate (TPR), also known as *Sensitivity* or *Recall*, is the ratio of correctly predicted positive cases (fraudulent transactions in the context of credit card fraud detection) to the actual positives.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F-measure (F1-score) is the harmonic mean of precision and recall, used to evaluate classification performance.

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

True Negative Rate (TNR), referred to as *Specificity*, measures the proportion of true negatives (legitimate transactions) correctly identified.

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

False Positive Rate (FPR) indicates the rate at which legitimate transactions are incorrectly flagged as fraud.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Receiver Operating Characteristics (ROC) graph plots TPR against FPR at various thresholds.

Area Under the Curve (AUC) indicates the classifier's ability to distinguish between classes, with higher values implying better prediction.

Matthews Correlation Coefficient (MCC) correlates observed and predicted classifications, ranging from -1 (completely incorrect) to +1 (perfect prediction).

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}$$

C Implementation

For implementing our experiments, we mainly use the Scikit-learn library for the baseline models and the framework PyTorch for the deep learning models. We run the experiments on one machine with an Intel Core i9-9900K CPU and

one NVIDIA GeForce RTX 3080 Ti GPU. CUDA[7] is set up and run for GPU acceleration. To make our results as reproducible as possible, we use the same random seed for all experiments. We use Weights & Biases[8] to track our experiments. Used hyperparameters can be found in the Appendix D.

D Hyperparameters

For each of the proposed DL architectures, we performed a hyperparameter search to find the best configuration. In the following, we list the different values we tried for every hyperparameter. We have highlighted in bold the parameters that yielded the best results.

- Validate size = 0.1 / 0.2
- Batch size = 128 / 256
- Epochs = 100 / 200
- Learning rate = 1e-2 / 1e-3 / 1e-4
- Patience = 5 / 10 / 20
- Delta (minimum change) = 0.01 / 0.001

The following are only relevant for autoencoders:

- Number of layers = **4** / 6 / 8
- Bottleneck size = 1 / 2 / 4 / 6 / **8** / 10
- Activation function = **'ELU'** / 'ReLU' / 'Tanh'
- Loss function = **'MSE'** / 'BCE'
- Noise factor = 0.5 (For DAE only)

The following are only relevant for the OC-NN:

- ν parameter = 0.0001 / 0.0005 / 0.001 / 0.005 / **0.01**
- r parameter = 1
- Hidden layer size = 8 / **12** / 16 / 20

E Additional results and plots

	Model	Precision	Recall	F1-Score	ROC-AUC	MCC
Baselines	Naive	0.5	1.0	0.66	0.5	0
	K-Means (k=2)	0.257	0.237	0.247	–	-0.451
	OC-SVM	0.579	**0.684**	**0.627**	0.617	0.19
	SGD OC-SVM	0.077	0.017	0.028	0.21	-0.299
	Isolation Forest	0.655	0.192	0.297	0.628	0.129
	kNN distance	**0.866**	0.369	0.518	**0.827**	**0.381**
	Mahalanobis Distance	0.604	0.077	0.136	0.601	0.054
	GMM	0.675	0.057	0.098	0.66	0.069
	KDE	0.662	0.25	0.363	0.635	0.157
DL Model	OC-NN	0.003	0	0	0.214	-0.08

Table 2: Performance of baseline models and DL Models in the hybrid approach

[7] https://developer.nvidia.com/cuda-toolkit

[8] https://wandb.ai

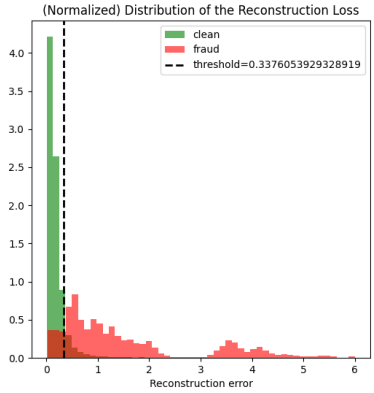

Figure 4: Autoencoder's reconstruction loss distribution of normal and anomaly data on the test set.

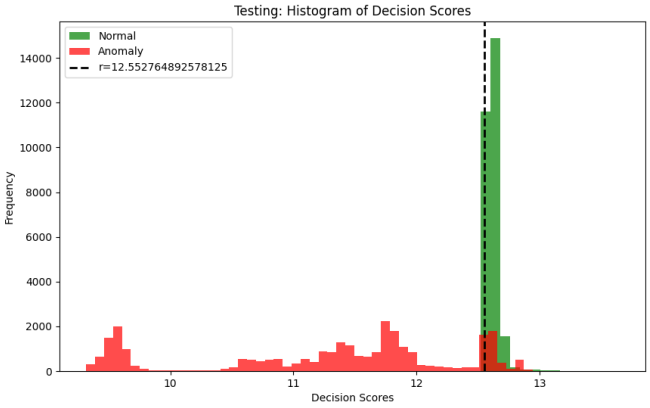

Figure 5: OC-NN's Anomaly score distribution of normal and anomaly data on the test set.

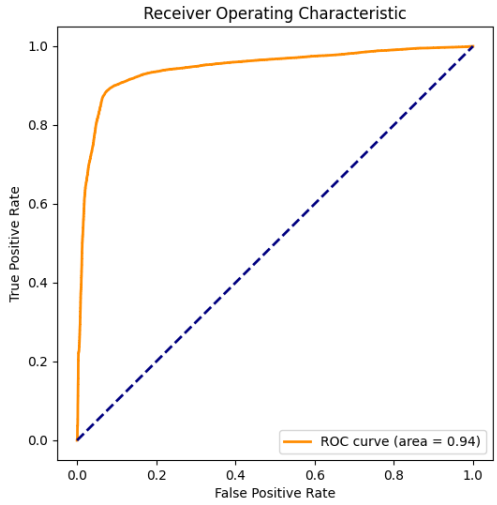

Figure 6: Autoencoder's ROC curve on the test set.

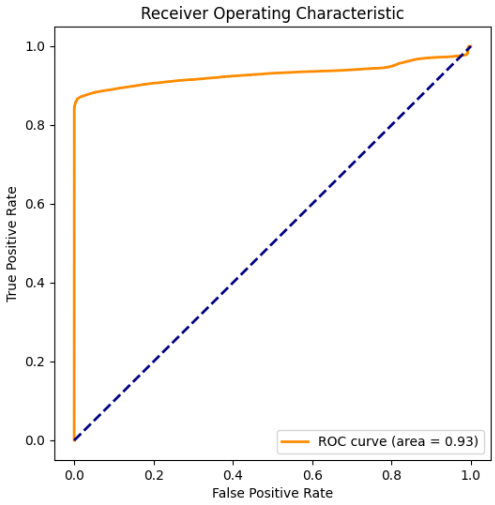

Figure 7: OC-NN's ROC curve on the test set.

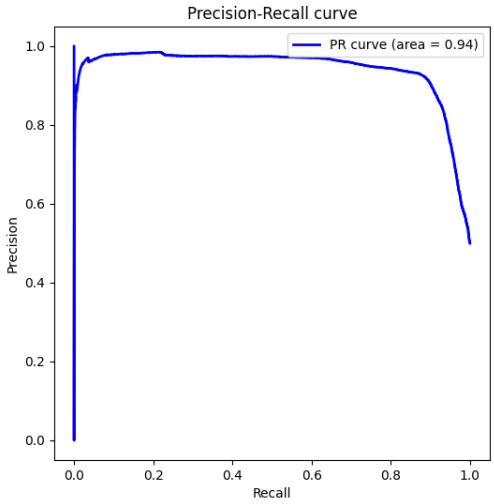

Figure 8: Autoencoder's PR curve on the test set.

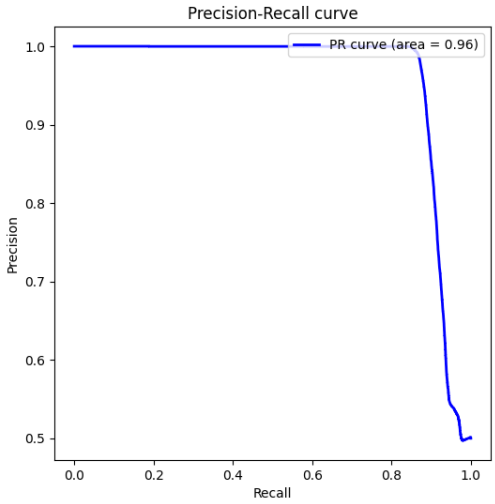

Figure 9: OC-NN's PR curve on the test set.