DevNet Reproducibility Report

Author: Fiona Zhang, Julia Jiang

Task Description

This report presents a reproducibility study and performance evaluation of an anomaly detection model known as devNet, using the creditcardfraud_normalized.csv dataset. The dataset contains normalized credit card transaction data with a significant class imbalance between legitimate and fraudulent activities. By applying a series of controlled data preparation techniques—including stratified splitting, anomaly control, and noise injection—the study aims to accurately assess devNet's ability to detect anomalies while ensuring that the experimental results are consistent and reproducible.

Data Understanding

The dataset(creditcardfraud_normalized.csv) under investigation comprises 284,807 rows and 30 columns. It includes 28 anonymized features labeled V1 through V28, an additional feature labeled Amount, and a target variable named class. An initial analysis of the target variable reveals a significant imbalance, with 284,315 instances of class 0 (99.83%) and only 492 instances of class 1 (0.17%). The dataset also contains 9,146 duplicate rows; however, no missing values were found across any of the columns. All non-target features are of a numerical type (float64), making them suitable for further correlation and statistical analyses.

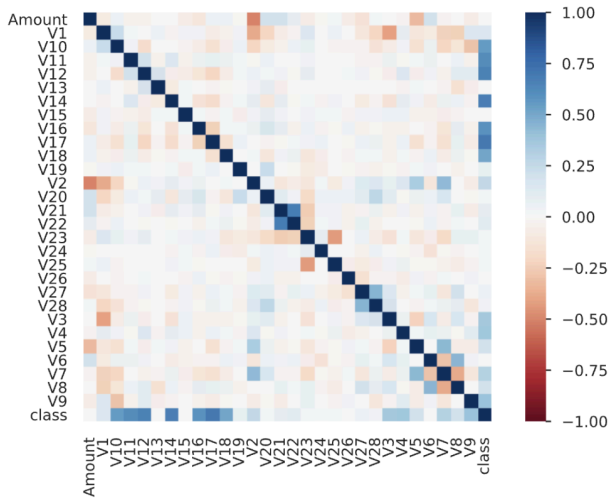
The table below summarizes the key statistics and properties of the dataset:

Metric	Value
Total Rows	284,807
Total Columns	30
Features (V1 - V28 & Amount)	29
Target Variable (class)	1
Class 0 Instances	284,315 (99.83%)

Class 1 Instances	492 (0.17%)
Duplicate Rows	9,146
Missing Values (All Columns)	0
Data Types (Non-target features)	float64
Data Type (Target feature)	int64

From the data understanding process, we do discover some interesting statistics:

1. Duplicate Rows: We do discover 9146 duplicate rows in the entire dataset, which is actually not a small number. Interestingly, the actual training process of devNet does not remove these duplicates.
2. Possible Data Leakage: By plotting the correlation graph for the dataset, we do discover features that have high correlation with the target column “class”, which can potentially become a target leakage in the training process.



Data Preparation and Model Evaluation

Data Preparation

- **Stratified Split:**

  The normalized dataset is first split into training (80%) and development (20%) sets using a stratified method and a fixed random state (42). This maintains the original imbalance between the classes.

- **Anomaly Control:**

  To ensure consistency, if there are more than 30 anomaly samples (class 1) in the training set, the extra anomalies are randomly removed using the fixed random state.

- **Noise Injection:**

  Based on a 2% contamination rate, noise samples are generated using the appropriate noise injection function (for dense or sparse data from devNet). These samples, with labels set to 0, are then added to the training data to simulate a realistic environment.

Model Evaluation

- **Holdout Validation:**

  The evaluation framework uses holdout validation. The devNet model is trained on the adjusted training data and then tested on the untouched development set. This approach helps assess how well the model generalizes to unseen data.

- **Evaluation Metrics:**

  Model performance is measured using two key metrics: AUC-ROC (Area Under the Receiver Operating Characteristic Curve) and AUC-PR (Area Under the Precision-Recall Curve), which are well-suited for assessing performance on imbalanced datasets.

- **Reproducibility:**

  Using a fixed random state for splitting, anomaly control, and noise injection ensures that the entire process is reproducible. This consistency is key for comparing model performance across experiments.

Modeling

Reproduced Model

- **Parameter Setting:**

  The DevNet model is integrated into the pipeline as a *CustomDevNet* instance with parameters closely following those in the referenced paper. We use a "shallow" architecture (one hidden layer with 20 units), a batch size of 512, and process 20

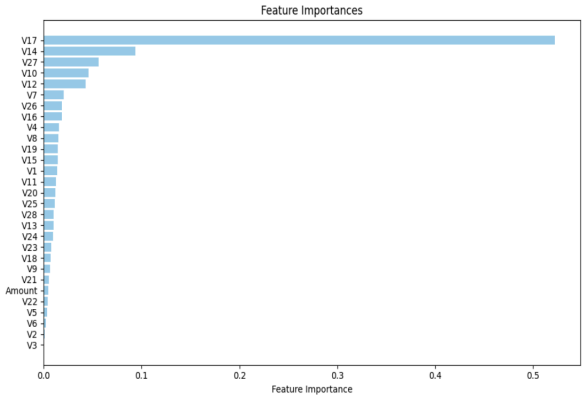
mini-batches per epoch over 50 epochs. A fixed random seed of 42 ensures reproducibility, and we use a dense data format (data_format=0) for our experiments.

- Comparison Model

  DevNet2 is an enhanced variant of the state of the art DEVNET (DevNet1) that incorporates an additional feature engineering step aimed at reducing redundancy in the input data. In this model, we first conduct a Spearman's correlation analysis among the features, and then delete specific features (for example, columns like 'V4', 'V10', 'V11', 'V12', 'V14') which are highly correlated with others. The motivation behind this approach is to remove noisy or redundant features that might obscure the true signal related to anomalies, potentially improving the model's ability to detect subtle outliers by focusing on more informative variables. The remainder of the model's architecture remains the same as in DevNet1—a shallow neural network with one hidden layer containing 20 units, using the same RMSprop optimizer and training regime.

  However, our experimental results show that DevNet2 did not lead to improvements over DevNet1. Despite the careful feature selection based on Spearman's correlation, which aimed to reduce redundancy and improve model efficiency, the removal of these features did not yield higher AUC-PR or AUC-ROC scores. In fact, the performance remains largely unchanged, indicating that in this particular dataset the deleted features may have contained useful information that the model needed to accurately distinguish between normal and anomalous instances. This outcome highlights the inherent challenge in feature selection: while removing collinear or redundant features can simplify the model, it may also inadvertently discard critical signals needed for robust anomaly detection.

- **Post-modeling (gini importance and possible data leakage)**



After fitting a decision tree model to the dataset, we analyzed the feature importances using Gini importance, which measures how each feature contributes to reducing impurity in the tree. The resulting bar plot highlights the most influential variables, with V17 standing out significantly—it alone accounts for more than half of the total importance, indicating that it plays a dominant role in the model's decision-making process. Other features like V14, V27, and V10 also show moderate contributions, while the majority of variables contribute minimally. Based on this analysis and the high accuracy we got from the modeling process, the dataset may have potential target leakage because of the significantly high

feature importance for V17 and the future model should try to focus on detecting data leakage and deleting duplicates in the dataset.

Results Comparison

f_1: Percentage of labeled anomalies in the training dataset

$$f_1 = \frac{Number\ of\ labeled\ anomalies\ in\ training}{Number\ of\ training\ samples}$$

f_2: Percentage of labeled anomalies relative to the total anomalies in the full dataset

$$f_2 = \frac{Number\ of\ labeled\ anomalies\ in\ training}{Total\ number\ of\ anomalies}$$

Data Characteristics					AUC-ROC			AUC-PR		
Model	#obj	f_1	f_2	D	Dummy	DevNet	iForest (baseline)	Dummy	DevNet	iForest (baseline)
Original	284,807	0.01%	6.10%	29	N/A	0.980±0.001	0.953±0.002	N/A	0.690±0.002	0.254±0.043
Reproduced	284,807	0.01%	6.10%	29	0.5000	0.9807	0.9545	0.0017	0.6906	0.2502
Comparison	275661	0.01%	6.34%	24	0.5000	0.864	N/A	0.0017	619	N/A

Scalability

In our scalability tests, we leveraged the principles behind SMOTE to generate synthetic samples for our minority class, and in doing so, effectively increased our dataset size. SMOTE works by taking a minority class instance, finding its five nearest neighbors within the same class, and then linearly interpolating between these points to create new synthetic examples. This technique doesn't simply duplicate data—it generates entirely new points in the feature space, which is key for testing how our model performs as the dataset scales up. By incrementally enlarging the dataset with these synthetic minority samples, we were able to observe changes in performance metrics like AUC-ROC and AUC-PR, giving us a clear picture of the model's scalability and robustness under varied data volumes.

Dataset Size Tested	AUC-ROC Score	AUC-PR Score
28,480	0.895	0.674

857,442	0.951	0.725
142,803	0.927	0.684
569,614	0.954	0.807
954,421	0.966	0.957
1,139,228	0.983	0.972
1,424,035	0.964	0.964
2,848,070	0.973	0.971

Strengths and Weakness

- **Strength**: DEVNET is highly reproducible for the task at hand, thanks to its robust design and flexible hyperparameters that allow fine-tuning during both training and testing. The model demonstrates notable resistance to noise—as evidenced by its consistently high AUC-ROC scores—even when noise is deliberately added during evaluation and the test sample size increases. This adaptability, coupled with adjustable settings that facilitate experimentation with various network configurations, ensures that DEVNET not only performs well under controlled conditions but also maintains solid predictive performance in diverse and noisy real-world scenarios. Additionally, based on the scalability test, the prediction for AUC-PR score largely increases with the increased size of the testing dataset.
- **Weakness**: One major limitation identified in the scalability report is that the test dataset is artificially generated, meaning it may not accurately reflect the intricacies of real-life data. The similarity between these synthetic test samples and the original dataset is uncertain, which could potentially lead to issues when the model is applied in practical settings. Additionally, the AUC-PR score for DEVNET isn't consistently high in development dataset, suggesting that while the model is effective at discriminating between normal and abnormal cases overall (as reflected by its AUC-ROC), it still faces challenges in detecting outliers—particularly those that are subtle or atypical in nature. Moreover, based on the post-modeling, there may be a potential target leakage that is causing this high accuracy.