Exploring GAN-Based Data Augmentation for Class-Imbalanced Credit Card Fraud Detection

Department of Artificial Intelligence

Special Topics in Artificial Intelligence

Dr. Yousef Sanjalawe

Jana Saleh Mahmoud Godieh 0223457

January 2026

Contents

Table of Figures

Problem Statement

Over the past decade, deep learning has become increasingly prominent in the field of financial fraud detection. The performance of these deep learning-based methods strongly relies on the availability and quality of training data. However, financial transaction datasets often suffer from severe class imbalance and limited fraud samples, posing a significant challenge when training reliable classification models. While a slight bias towards the dominant class may be tolerated in other domains, such bias is unacceptable in fraud detection, where false negatives result in direct financial losses and false positives erode customer trust and satisfaction. This concern is especially evident in credit card fraud detection, where fraudulent transactions represent less than 0.2% of all transactions, yet account for billions of dollars in annual losses globally. According to the 2025 Nilson Report, global payment card fraud losses reached approximately $33.83 billion in 2023, underscoring the significant economic impact of fraudulent transactions worldwide. Despite representing less than 0.2% of all transactions, such fraudulent events create severe class imbalance that hinders effective classifier training.

Recent advancements in generative models, particularly generative adversarial networks (GANs), have demonstrated substantial potential in generating synthetic tabular data, enhancing minority class representation, and improving classifier performance on imbalanced datasets. In this project, different GAN variants (VanillaGAN, WGAN-GP, and CGAN) are explored to balance the minority class in a credit card fraud detection dataset by generating synthetic fraudulent transactions. The effectiveness of this approach is evaluated by comparing the performance of a logistic regression classifier trained on the original imbalanced dataset and on balanced datasets using GAN-generated samples.

Description of Dataset & Imbalance Analysis

The dataset used in this project is the Credit Card Fraud Detection dataset, available publicly on Kaggle and originally collected and made available by the Machine Learning Group at Université Libre de Bruxelles (ULB) as part of a collaborative research project on fraud detection. The dataset contains 284,807 credit card transactions made by European cardholders over the span of two days in September 2013 and has been anonymized for privacy. The dataset consists of two classes: fraudulent transactions and legitimate transactions, representing a highly imbalanced real-world scenario where fraud cases constitute only 0.172% of all transactions (492 fraudulent transactions out of 284,807 total).

The dataset contains 30 numerical features:

- V1-V28: These are the results of a Principal Component Analysis (PCA) transformation applied to the original transaction features. Due to confidentiality and privacy constraints, the original variables and detailed background information cannot be disclosed. The PCA

components preserve the essential statistical structure of the data while ensuring anonymity.

- Time: The number of seconds between each transaction and the first transaction in the dataset
- Amount: The transaction amount (in the original currency before scaling). This feature was not included in the PCA transformation and is retained in its original form.

The severe class imbalance in this dataset (99.828% legitimate vs. 0.172% fraudulent) makes it particularly challenging for traditional classification algorithms, which tend to bias toward the majority class, thereby motivating the use of data augmentation techniques such as GAN-based synthetic sample generation.

To assess the class imbalance and understand the data distribution, multiple analyses and visualizations were performed, including class distribution plots, correlation analysis of features, and statistical summaries of transaction amounts for both classes.

Class Distribution Analysis

The Credit Card Fraud Detection dataset exhibits severe class imbalance. As illustrated in Figure 1, the dataset contains 284,315 legitimate transactions (Class 0) compared to only 492 fraudulent transactions (Class 1), resulting in an imbalance ratio of 578:1 or 99.83% to 0.17%. The log-scale visualization clearly demonstrates the magnitude of this disparity, where fraudulent transactions are barely visible on a linear scale.

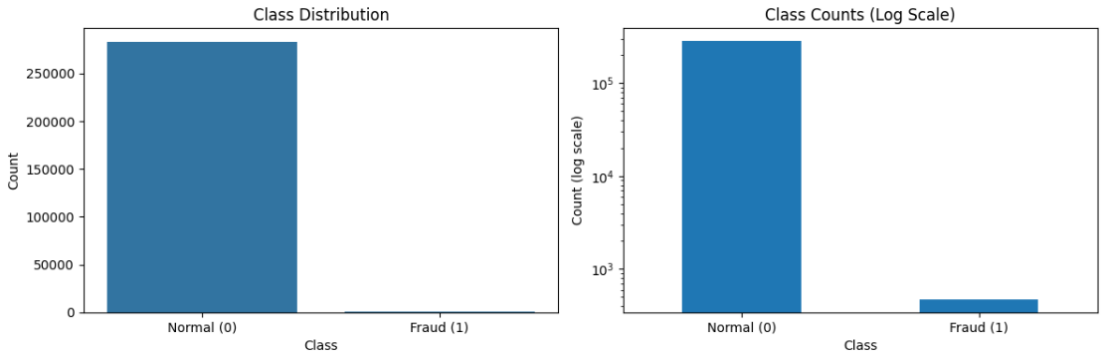

Figure 1- Class Distribution bar plot

This extreme imbalance strongly biases conventional classifiers toward the majority class, often producing misleadingly high accuracy while failing to detect fraud. This motivates the use of GAN-based synthetic data generation to balance the training set and improve minority-class learning.

Train-Validation-Test Split

The dataset was split into training (64%), validation (16%), and test (20%) sets using stratified sampling to preserve the original fraud ratio (0.172%) across all subsets. This resulted in 181,584 training samples (302 fraud), 45,396 validation samples (76 fraud), and 56,746 test samples (95 fraud).

Figure 2 shows that the severe class imbalance is consistently maintained across all splits on a logarithmic scale. Importantly, only 302 fraudulent transactions are available in the training set, which are used to train the GAN models (VanillaGAN, WGAN-GP, and CGAN) to generate synthetic fraud samples. The test set remains untouched and is used exclusively for unbiased performance evaluation.

Figure 2 - Class Distribution across split

Visual Analysis of Training split Imbalance

Figure 3 further illustrates the extent of class imbalance, each square represents approximately 0.1% of the data. Almost the entire grid corresponds to legitimate transactions (blue), while fraudulent transactions (orange) appear as only a few isolated blocks. This highlights how rare fraud cases are in practice

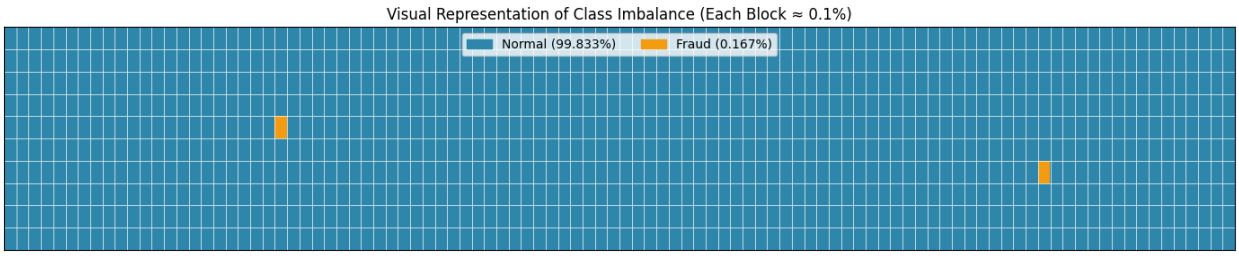

Figure 3 - waffle plot

GAN architectures explored and Training details

Vanilla GAN

The Vanilla GAN utilizes a fully connected architecture for both generator and discriminator networks. The generator maps a 64-dimensional latent vector sampled from a standard normal distribution through a series of linear layers (64→128→256→30) with LeakyReLU activations (negative slope of 0.2), producing synthetic fraudulent transaction features matching the 30-dimensional feature space of the preprocessed dataset. The discriminator implements a symmetric architecture in reverse (30→256→128→1), processing input transactions through linear layers with LeakyReLU activations and dropout regularization (p=0.2) to prevent overfitting, before outputting a single logit value representing the authenticity score. Training follows the standard minimax game formulation with binary cross-entropy with logits loss (BCEWithLogitsLoss), where the discriminator is trained to distinguish real fraudulent transactions from generated synthetic samples while the generator learns to fool the discriminator into classifying its outputs as real. Both networks are optimized using Adam with a learning rate of 2e-4 and β=(0.5, 0.999), performing alternating updates over 300 epochs with a batch size of 128. Training was conducted exclusively on the 302 fraudulent transactions from the training set, enabling the generator to learn the underlying distribution of fraud patterns and produce synthetic samples for dataset balancing.

WGAN-GP (Wasserstein GAN with Gradient Penalty)

WGAN-GP fundamentally differs from traditional GANs by replacing the discriminator with a critic that approximates the Wasserstein distance between real and generated distributions, addressing common training instabilities such as mode collapse and vanishing gradients. The architecture employs fully connected layers with the critic processing transactions through linear layers (30→256→128→1) with LeakyReLU activations (negative slope of 0.2), producing unbounded scores rather than probabilities to directly estimate the Wasserstein distance. Unlike Vanilla GAN, the critic architecture notably excludes dropout regularization, as this would interfere with the gradient penalty computation that requires stable gradient flow. The generator maintains an identical architecture to Vanilla GAN (64→128→256→30) with LeakyReLU activations. The gradient penalty term enforces the Lipschitz constraint by penalizing deviations of the critic's gradient norm from 1 on interpolated samples between real and fake transactions, using $\lambda\_gp=10$. Training follows the Wasserstein objective where the critic maximizes the difference between scores for real fraudulent transactions and generated samples (Wasserstein

distance) while the generator minimizes the critic's score for its outputs. Following the original WGAN-GP specification, the critic is trained for 5 iterations per generator update (n_critic=5) to ensure the critic remains near optimal, using Adam optimizer with a lower learning rate of 1e-4 and modified beta values (β= (0.0, 0.9)) for both networks 300 epochs with a batch size of 128.

CGAN (Conditional GAN)

CGAN extends the standard GAN by conditioning both the generator and discriminator on class labels, enabling controlled generation of either fraudulent or legitimate transactions. Class information is incorporated through an 8-dimensional embedding of the binary labels (0 = normal, 1 = fraud), which is concatenated with the network inputs. The generator processes the combined latent–label vector ((64+8) →128→256→30) with LeakyReLU activations to produce class-conditioned synthetic transactions, while the discriminator concatenates input features with label embeddings and applies a mirrored architecture ((30+8)→256→128→1) with LeakyReLU and dropout (p=0.2) to evaluate authenticity.

Unlike Vanilla GAN, CGAN is trained on the full imbalanced training set (181,584 samples), allowing it to learn the joint distribution of features and labels. During training, class labels are randomly sampled (Bernoulli p=0.5) to enable balanced conditional generation. Training uses binary cross-entropy with logits and Adam optimization (lr = 2e-4, β = (0.5, 0.999)) over 300 epochs with a batch size of 256, enabling targeted synthesis of fraudulent transactions while preserving class distinctions.

The table below presents the hyperparameter configuration for all three GAN variants. To ensure a fair comparison, most parameters were kept consistent across models, with differences only where required by the underlying GAN formulation.

Hyperparameter	Vanilla Gan	WGAN-GP	CGAN
Architecture	Fully connected	Fully connected	Fully connected
Epochs	300	300	300
Batch size	128	128	256
Learning rate	2e-4	1e-4	2e-4
Betas	0.5, 0.999	0.0,0.9	0.5,0.999
Optimizer	Adam	Adam	Adam
Latent Dimension(z)	64	64	64
Loss Function	BCEWithLogitsLoss	Wasserstein and GP	BCEWithLogitsLoss

Generated Synthetic data examples

To qualitatively assess the realism of the generated samples, several rows of synthetic fraudulent transactions produced by each GAN variant are shown in the tables below. Statistical comparisons between real and synthetic fraud samples show that both WGAN-GP and CGAN preserve the underlying feature distributions substantially better than Vanilla GAN, which exhibits larger deviations across multiple features .WGAN-GP reduces extreme deviations in high-variance features such as V8, V14, and V17 by enforcing smoothness via the gradient penalty, while CGAN achieves tighter mean matching on several features (for example V28, V25, V15, and Time) through label-conditioned generation. These trends indicate that WGAN-GP emphasizes distributional stability, whereas CGAN provides more precise class-specific alignment, making both suitable for improving minority-class learning.

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.463783	0.004521	2.047815	-3.076808	2.155271	-0.190921	-1.264514	0.123177	-0.635980
1	0.488752	0.109879	1.655250	-2.212796	1.388500	-0.611897	-0.826471	-0.014084	-0.524077
2	0.466458	-0.690918	1.783148	-2.558172	1.653494	-0.929845	-1.345787	-0.881322	0.153154
3	0.802118	-0.931706	2.575675	-3.009526	2.319309	-0.588111	-1.961424	-0.712137	0.426928
4	1.423622	-1.309061	3.654131	-4.923864	2.947370	-1.704619	-2.570253	-1.334002	0.524826

Figure 4-Vanilla GAN Synthetic Fraudulent data example

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	-0.331809	-3.493893	3.747186	-5.745718	3.118732	-3.573433	-1.519571	-6.282854	5.156950
1	0.129185	-1.386776	0.789325	-1.605813	1.809296	-1.762698	-0.107188	-3.080119	3.306684
2	-0.099762	-1.129274	0.630084	-2.142265	2.034606	-1.388336	-0.818257	-3.350545	3.139929
3	-0.339302	-1.704015	2.534148	-3.783882	2.773828	-3.259084	-0.836146	-5.297043	3.235386
4	0.219102	-2.542452	1.644755	-2.571228	2.549278	-2.062573	-0.620962	-5.229233	1.122496

Figure 5-WGAN Synthetic Fraudulent Data Example

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.227098	1.233437	-0.166695	-1.671932	0.960490	-0.713580	-1.499798	0.668628	-0.612490
1	0.195501	1.526864	-0.099213	-2.210153	0.975310	0.512261	-1.423173	0.956348	-1.269690
2	0.523822	1.302577	-0.233662	-1.844197	1.114401	-0.441930	-1.478934	0.669472	-0.811698
3	0.141987	1.436360	-0.155032	-2.151464	1.089632	0.172800	-1.408144	0.995371	-1.200090
4	0.299174	1.287083	-0.337369	-1.856281	1.150071	-0.453050	-1.370547	0.741326	-0.827340

Figure 6- CGAN Synthetic Fraudulent data Example

Classifier Setup and Evaluation

Classifier Implementation

A logistic regression classifier was used to evaluate the impact of GAN-based data augmentation on fraud detection performance. The model was implemented using scikit-learn's LogisticRegression with max_iter = 2000 and class_weight = 'balanced', ensuring equal importance was assigned to both classes despite the severe imbalance. The same classifier configuration was used across all experiments to enable a fair comparison between training scenarios.

Training Scenarios

Four training scenarios were evaluated:

- Original Imbalanced data: Baseline training on 181,282 legitimate and 302 fraudulent transactions (with a ratio of 600:1)
- VanillaGAN-balanced data: Training set balanced by generating 180,980 synthetic fraud samples, achieving 181,282 samples per class
- WGAN-GP-balanced data: Similar balancing using WGAN-GP-generated synthetic fraud samples
- CGAN-balanced data: Balanced using CGAN-generated samples, with CGAN trained on the full dataset with label conditioning

Evaluation Methodology

All scenarios were evaluated on the same held-out test set (56,651 legitimate, 95 fraudulent) and a fixed threshold of 0.5 for all scenarios to assess performance on genuine, unseen fraud patterns. Performance metrics included:

- **ROC-AUC**: Overall discrimination capability across all thresholds
- **PR-AUC**: Minority class performance, more informative for imbalanced data
- **Precision/Recall/F1 (Fraud)**: Fraud detection effectiveness, balancing false alarms vs. missed frauds
- **Confusion Matrix**: Detailed breakdown of TP, FP, TN, FN for error analysis

Results & Comparisons

Overall performance

	Scenario	ROC-AUC	PR-AUC	Precision (Fraud)	Recall (Fraud)	F1 (Fraud)	TP	FP	FN	TN
2	WGAN-GP-balanced	0.977393	0.706658	0.752475	0.800000	0.775510	76	25	19	56626
1	VanillaGAN-balanced	0.920119	0.676163	0.782609	0.757895	0.770053	72	20	23	56631
0	Original (Imbalanced)	0.980755	0.668344	0.065364	0.915789	0.122020	87	1244	8	55407
3	CGAN-balanced	0.964079	0.641388	0.773810	0.684211	0.726257	65	19	30	56632

The baseline classifier trained on the original imbalanced data achieved a high ROC-AUC of 0.981, but this was misleading: although it detected most frauds (recall = 0.916), its precision was extremely low (0.065), producing 1,244 false alarms for only 87 detected frauds. This confirms that severe class imbalance causes the classifier to over-predict fraud, making it impractical in real-world deployment.

After GAN-based balancing, all three synthetic-data approaches dramatically improved fraud precision while maintaining high recall. VanillaGAN-balanced data increased fraud precision to 0.783 while preserving recall at 0.758, raising the fraud F1-score from 0.12 to 0.77. WGAN-GP achieved the strongest overall performance, reaching the highest PR-AUC (0.706) and the best fraud F1-score (0.776), indicating superior minority-class modeling. CGAN also substantially improved over the baseline, achieving a fraud F1-score of 0.726, but slightly underperformed WGAN-GP and VanillaGAN in recall and PR-AUC.

Fraud Detection

Because fraud detection is a rare-event problem, PR-AUC and recall are more meaningful than ROC-AUC. WGAN-GP produced the highest PR-AUC (0.706), demonstrating that its synthetic samples most effectively improved the classifier's ability to identify fraud with fewer false positives. WGAN-GP also achieved the highest fraud recall (0.80), detecting 76 out of 95 fraud cases while keeping false alarms relatively low (25).

VanillaGAN performed similarly, detecting 72 fraud cases with slightly fewer false positives (20), but its recall (0.758) and PR-AUC (0.676) were marginally lower than WGAN-GP. CGAN

showed weaker fraud recall (0.684), missing more fraud cases despite maintaining good precision.

Training stability

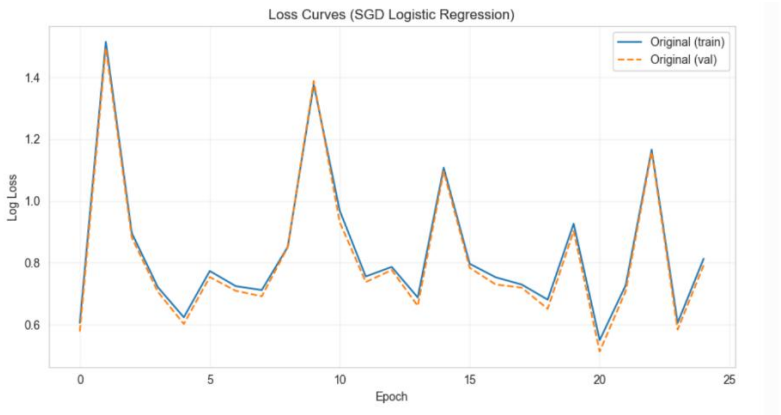

Figure 7- Original data distribution loss curves

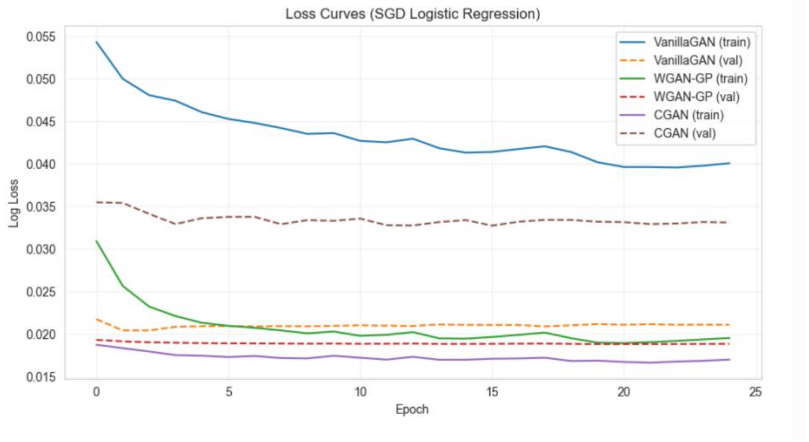

Figure 8-Balanced dataset loss curves

The learning-curve plots (Figures 7 and 8) further support these results. GAN-balanced datasets led to smoother, lower validation loss compared to the original imbalanced case, indicating better generalization. Among GANs, WGAN-GP exhibited the most stable convergence, consistent with its theoretical advantages against mode collapse and training instability.

Error analysis using confusion matrices

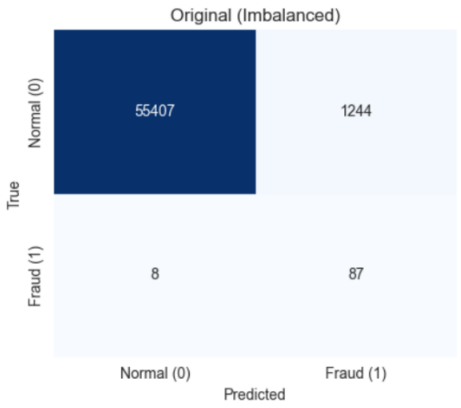

Figure 9-Original data confusion matrix

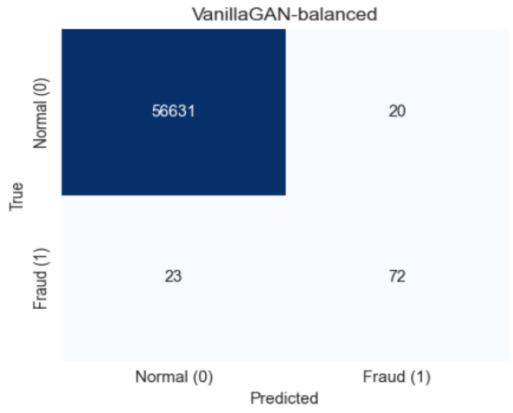

Figure 10- Vanilla GAN balanced data confusion matrix

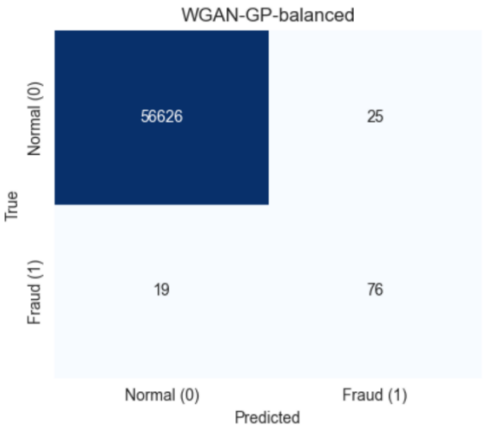

Figure 11-WGAN Balanced Data Confusion Matrix

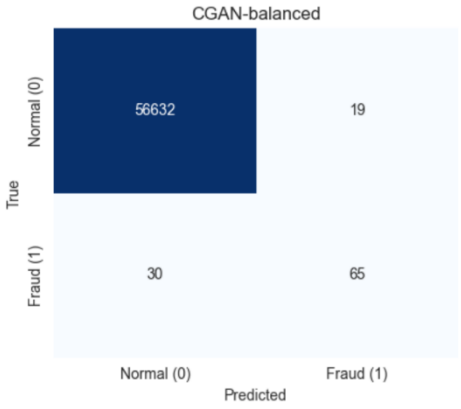

Figure 12-CGAN balanced confusion matrix

The confusion matrices (Figures 9–12) highlight how GAN-based balancing reshapes the classifier's behavior. In the original imbalanced setting, the model flagged a very large number of legitimate transactions as fraud (1,244 false positives), severely limiting usability. After GAN balancing, false positives dropped by more than **98%**, while true fraud detections remained high.

WGAN-GP produced the most balanced trade-off, combining strong fraud detection (76 TP) with controlled false alarms (25 FP). This confirms that the Wasserstein loss and gradient penalty helped generate more realistic fraud samples that generalize better to unseen test data.

Observations and Conclusions

The results demonstrate the effectiveness of GAN-based data augmentation for imbalanced fraud detection. Although the baseline model achieved a high ROC-AUC (98.1%), it suffered from extremely low precision (6.5%) and excessive false alarms, making it impractical for real-world use. All GAN-based methods substantially improved minority-class performance, increasing precision to 75–78%, recall to 68–80%, and reducing false positives by over 98%. Among them, WGAN-GP performed best, achieving the highest PR-AUC (0.707) and F1-score (0.776), indicating superior synthetic data quality. Its Wasserstein objective and gradient penalty produced more realistic fraud samples that generalized better to unseen data. Training-loss stability further supported this, as GAN-balanced models, particularly WGAN-GP, exhibited smoother convergence than the highly unstable imbalanced baseline.

This project successfully demonstrated that GAN-based data augmentation effectively addresses severe class imbalance in credit card fraud detection:

- Effectiveness: GAN balancing improved fraud precision from 6.5% to 75-78% while maintaining 68-80% recall, achieving practical deployment thresholds
- Best approach: WGAN-GP achieved optimal precision-recall balance and training stability, making it the recommended variant
- Practical impact: 98% reduction in false positives while detecting 80% of frauds aligns with real-world requirements for minimizing both financial losses and customer friction

References

The Nilson Report (2025). “Payment Card Fraud Losses Worldwide Approach $34 Billion.” NilsonReport.com. Retrieved from https://nilsonreport.com/newsletters/1298/ (accessed January 2026).