Electronics and information technologies, 2025, 32, 101–120
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC 004.94

DEVELOPMENT OF A DEEP LEARNING MODEL FOR FRAUD DETECTION

***Sergiy Sveleba*[1]*, *Ivan Katerynchuk*[1], *Ivan Kunyo*[1], *Ihor Polovynko*, *Yaroslav Shmyhelskyy*[1], *Ostap Sumyliak*[2]**

*[1]Ivan Franko National University of Lviv,
107 Tarnavsky St., UA–79017 Lviv, Ukraine*
*[2]Private Higher Education Establishment "European University",
16B, Academician Vernadsky Boulevard, Kyiv, Ukraine*

Sveleba, S., Katerynchuk, I., Kunyo, I., Polovynko, I., Shmyhelskyy, Ya., & Sumyliak, O. (2025). Development of a Deep Learning Model for Fraud Detection. *Electronics and Information Technologies*, 32. 101–120. https://doi.org/10.30970/eli.32.7

ABSTRACT

Background. The rapid growth of electronic payments has intensified fraudulent activity, requiring adaptive anomaly detection methods. Traditional rule-based approaches lack flexibility and fail to generalize to previously unseen attacks. In contrast, unsupervised deep learning models, particularly autoencoders, can learn intrinsic data representations and detect anomalies without labeled attack samples. This study evaluates three unsupervised architectures – Autoencoder with Gaussian Mixture Model (AEGMM), Variational Autoencoder with Gaussian Mixture Model (VAEGMM), and a Deep Autoencoder – for network anomaly detection.

Materials and Methods. Experiments were conducted using the KDD'99 (10%) benchmark dataset. Categorical features were transformed using one-hot encoding, while numerical features were standardized. All models were trained exclusively on normal traffic samples following a one-class learning paradigm. The experimental pipeline included preprocessing, model implementation in Python using TensorFlow and the Alibi Detect framework, percentile-based threshold calibration, and evaluation using accuracy, precision, recall, F1-score, and confusion matrices.

Results. AEGMM achieved the highest performance with an F1-score of 0.9936 and an accuracy of 0.9908, demonstrating near-perfect separation between normal and malicious samples. VAEGMM reached an F1-score of 0.9751, showing stable convergence but slightly reduced accuracy due to the stochastic latent space. The Deep Autoencoder achieved approximately 97.5% accuracy, confirming the effectiveness of reconstruction-based methods without probabilistic density estimation. The optimal anomaly threshold, defined at the 99th percentile of reconstruction or density scores, ensured reliable discrimination between normal and attack states.

Conclusion. Autoencoder-based unsupervised models are effective for anomaly detection in large, imbalanced tabular datasets. AEGMM outperformed alternative architectures due to its stable latent representation and deterministic optimization. The proposed approach is suitable for financial fraud detection, cybersecurity monitoring, and industrial anomaly detection. Future work will explore transformer-based models and Explainable AI to improve robustness and interpretability.

***Keywords*:** artificial intelligence; deep learning; autoencoder; Gaussian mixture model; financial fraud; cybersecurity

INTRODUCTION

The use of Artificial Intelligence (AI) for detecting suspicious banking transactions has become increasingly relevant in the modern world of financial technologies. The application of intelligent systems enables effective identification and prevention of financial crimes such as money laundering, fraud, and other manipulative activities. AI is capable of analyzing vast amounts of transactional data in real time, efficiently identifying anomalies and atypical behavioral patterns. This allows banks to promptly respond to potentially fraudulent activities, minimizing financial risks for both clients and institutions. Most modern banking systems have already integrated AI-based tools to enhance the security of financial operations. Given the continuous technological progress, the role of AI in the banking sector will continue to grow, ensuring more reliable and efficient protection of financial systems.

Fraud is one of the most widespread phenomena in finance, encompassing such crimes as money laundering, falsification of financial statements, phishing, cyber fraud, and credit card fraud. The rapid expansion of digital banking has led to the emergence of new types of digital fraud. As the number of digital technology users increases, the frequency and complexity of fraudulent scenarios are expected to rise, making traditional rule-based systems ineffective and difficult to scale. Furthermore, a significant issue lies in false positives – legitimate transactions mistakenly classified as fraudulent – which result in customer dissatisfaction and substantial financial losses for banks.

A study by Chiabanu (2020) revealed that approximately 25% of customers whose legitimate transactions were incorrectly declined subsequently moved to competing institutions. Among clients aged 18–24, this percentage increased to 36%, and among those aged 25–34 — to 31%. These findings demonstrate the urgent need to improve fraud detection systems.

The financial, banking, and fintech sectors face various fraud typologies each year, which can be broadly classified into:

- Digital fraud,
- Physical attacks,
- Internal collusion,
- Violations of the “four-eyes principle.”

The first two types correspond to external threats, while the latter two arise from internal misuse or employee-related schemes. Digital fraud encompasses a wide range of illicit online activities, where Machine Learning (ML) and Deep Learning (DL) serve as key tools for counteraction.

Problem Statement

With the advancement of technologies, fraudsters continuously evolve their methods, rendering traditional rule-based systems inefficient. Such systems require frequent updates and fail to adapt quickly to emerging fraudulent patterns. Hence, there is a growing necessity to develop adaptive models capable of autonomously detecting new forms of fraudulent behavior based on historical data.

According to the Nielsen Report (2021), losses caused by fraudulent activities involving payment cards amounted to USD 32.34 billion, a 14% increase compared to the previous year. The growing volume of digital transactions via cards and mobile devices has intensified the demand for robust and scalable fraud detection systems. The emergence of AI has opened vast opportunities for designing such intelligent solutions. Industry leaders — Google, Facebook, Apple, Amazon, and Netflix — are actively deploying AI-driven tools in their internal financial processes, setting high standards for analytical performance and security in the global financial sector.

Literature Review

Systematic reviews covering the period 2019–2024/25 highlight a paradigm shift from traditional tabular ML approaches toward deep architectures that incorporate temporal, graph-based, and contextual dependencies in transaction data. Increasing attention is being paid to issues such as class imbalance, concept drift, model stability, and interpretability.

For financial services, graph-based approaches (modeling relationships such as *client–card–device–merchant*), sequential or transformer-based models for transaction streams, and hybrid architectures (deep representations combined with gradient boosting) have become the de facto standard. The main research focus is on reducing false positives and improving the robustness of models in real-world financial scenarios.

Architectures Demonstrating Practical Effectiveness

1. Sequential Models. Architectures such as RNN, LSTM, Temporal-CNN, and Transformers are employed to model temporal patterns in customer or card behavior. Self-supervised contrastive pretraining significantly enhances model performance in *cold-start* and high-class-imbalance conditions [1].
2. Graph Neural Networks (GNN) Relationships of the type *account* ↔ *device* ↔ *IP* ↔ *merchant* naturally form a graph-structured dataset. Models like GCN, GAT, and encoder–decoder GNNs consistently outperform tabular approaches, particularly in identifying organized fraud schemes and fraud rings [2].
3. Autoencoders, VAE, and GAN. These architectures are effective as unsupervised / one-class models for anomaly detection, feature extraction, and the discovery of novel fraud scenarios — especially in cases where labeled data is scarce or delayed [3].
4. Concept Drift and Streaming Adaptation. Real-world payment streams exhibit high variability (changing attacker strategies, seasonality, marketing campaigns). Studies on concept drift propose detectors such as ADWIN, DDM, and one-class classifiers, along with dynamic model updating and threshold adaptation strategies. In financial applications, the best results are achieved through a combination of online learning, active labeling, and periodic retraining [4].
5. Explainable AI (XAI) and Compliance Requirements. Most banking institutions +Key focus areas include:
   – Local explanations for escalation cases,
   – Attribution stability,
   – Generation of reason codes for analysts. Emerging research emphasizes user-centered XAI protocols and their integration into federated learning frameworks [4].
6. Confidentiality and Federated Learning (FL). In banking consortia, Federated Learning enables collaborative model training without sharing raw data. These architectures often include XAI layers to ensure auditability and regulatory compliance [5].
7. Public Datasets and Benchmarks
   – IEEE-CIS (Kaggle) - a large e-commerce dataset with the binary target isFraud; widely used for tabular, sequential, and hybrid models [6].
   – CreditCard (ULB) - a classical dataset with PCA features and strong class imbalance, suitable for rapid prototyping [6].
   – PaySim (synthetic mobile money) - simulates large-scale transaction systems under class imbalance [6].
   – Elliptic (Bitcoin AML) - a transaction graph labeled *licit / illicit*; a key benchmark for GNN- and AML-oriented research, with expanded 2024–2025 versions [7].
8. Technical Design Aspects

– Class imbalance: handled using focal loss, cost-sensitive learning, threshold calibration, and advanced oversampling heuristics tailored for fraud detection [8].
– Evaluation metrics: AUC-PR, precision@k, FPR@TPR, and expected cost are preferred over traditional accuracy due to extreme imbalance (e.g., only 3–4% fraudulent transactions in IEEE-CIS) [9].
– Online inference: requires millisecond-level latency, streaming feature engineering (*rolling windows*), feature stores, and continuous monitoring of drift with scheduled retraining [2].

9. Architectural Trends Summary. Graph-based models (for card, e-commerce, and crypto scenarios) - such as encoder–decoder GNNs, GCN/GAT architectures on the Elliptic dataset, and industrial-scale GNN blueprints - currently define the standard for fraud detection. Self-supervised contrastive pretraining significantly improves performance under limited labeling and pattern variability [7]. Concept drift handling is a mandatory component of modern systems, with integrated monitoring and update policies. Finally, XAI and Federated Learning are not optional additions but essential requirements for practical deployment, ensuring transparency, compliance, and trust among stakeholders [12].

Autoencoder

An Autoencoder is a type of neural network trained to reproduce its input at the output through a narrow bottleneck layer – the latent space, where it learns to compress (encode) only the most salient information.

Structure:

1. Encoder – transforms the input vector *x* into a lower-dimensional hidden representation *z*:

$$z = f_{\text{enc}}(x) = \sigma(W_1 x + b_1)$$

2. Bottleneck (Latent Space) – the narrowest part of the network, storing essential information about the input.
3. Decoder – reconstructs $\hat{x}$ from the encoded representation *z*:

$$\hat{x} = f_{\text{dec}}(z) = \sigma(W_2 z + b_2)$$

Loss Functions: Autoencoders aim to minimize the difference between the input and reconstructed data, typically using:

- Mean Squared Error (MSE): $L(x, \hat{x}) = \|x - \hat{x}\|^2$
- Binary Cross–Entropy (BCE): $L(x, \hat{x}) = -\sum_i [x_i \, log(\hat{x}_i) + (1 - x_i) \, log(1 - \hat{x}_i)]$

Types:

- *Shallow Autoencoder* – single hidden layer.
- *Deep Autoencoder* – multiple hidden layers for complex data compression.
- *Sparse Autoencoder* – uses regularization to keep most neurons inactive.
- *Denoising Autoencoder* – reconstructs clean inputs from corrupted versions.
- *Variational Autoencoder (VAE)* – introduces probabilistic encoding through μ and σ distributions, enabling generative modeling.

Training Pipeline:

1. Data preprocessing – normalization, scaling, and optional noise addition.
2. Architectural design – selection of layer sizes for encoder, latent space, and decoder.
3. Loss and optimizer selection – typically *Adam* or *RMSprop*.
4. Training – input serves simultaneously as target $(x \to \hat{x})$.
5. Evaluation – reconstruction error analysis and visual comparison of results.

MATERIALS AND METHODS

Data Source. For our experiments, we used the open dataset available on Kaggle: [12]. This benchmark corpus is a standard testbed for intrusion detection in computer networks modeled in a military environment. The anomaly detector (decoder model) is designed to identify intrusion attempts using TCP dump data from a local area network (LAN) that emulates a typical U.S. Air Force infrastructure.

Data Description. Each connection is a sequence of TCP packets with a defined start and end time, capturing data exchange between a source and a destination IP address under a specific protocol. Every connection is labeled as either “normal” or “attack.” The dataset includes four primary attack categories:

- DOS (Denial of Service): e.g., SYN flood.
- R2L (Remote to Local): unauthorized access from a remote machine (e.g., password guessing).
- U2R (User to Root): unauthorized escalation to superuser (root) privileges.
- Probe: reconnaissance, such as port or service scanning.

The corpus contains approximately five million connection records. Features are grouped into three classes:

1. Basic connection characteristics (e.g., connection duration);
2. Content features within a session (e.g., number of failed login attempts);
3. Traffic features over a 2-second window (e.g., number of connections to the same host as the current one).

Implementation. We prepared a Python script and a README to:

- download the KDD Cup 1999 (10%) subset from Kaggle;
- assemble a DataFrame;
- perform one-hot encoding of categorical variables (*protocol_type, service, flag*);
- produce two feature variants:
  – standardized (via StandardScaler),
  – normalized to [0, 1] (via MinMaxScaler);
- save the processed CSV files and the fitted scalers in *.pkl* format.

Variational Autoencoder (VAE) Construction. Because the original dataset is large, only a subset (≈10%) is used to train the VAE. The model is trained exclusively on normal (non-attack) samples; data are then standardized.

Data Preparation and Standardization. After preliminary selection, standardization and normalization are applied. For convenience, one may use a built-in fetch detector function, which stores pre-prepared models in a local directory and loads the detector; if the directory does not exist, it is created automatically.

During initialization, a warning indicates that an outlier threshold must be specified. This is set with infer threshold(), which takes a batch of instances and the parameter threshold perc indicating what proportion of the data is considered normal. For example, if ≈5% of the data are known outliers, that proportion can be supplied via perc outlier in create outlier batch(). The threshold can also be inferred solely from normal training data by setting threshold perc = 99 and adding a small safety margin above the inferred value.

Outlier Detection. A batch containing 10% outliers is then constructed and scored. After predicting outliers, model quality is assessed using the F1-score and the confusion matrix.

On the outlier plot, some outliers separate clearly while others lie closer to the normal region. In addition, an ROC curve can be drawn to illustrate detector performance across thresholds. An analysis of individual predictions (e.g., on X outlier) indicates that the feature srv count frequently contributes strongly to marked outliers.

Experiment Template for KDD’99 (10%). We prepared a ready-to-use template that ingests processed CSV files and demonstrates two anomaly-detection approaches for tabular data using Alibi Detect:

- AEGMM (Autoencoder + Gaussian Mixture Model),
- VAEGMM (Variational Autoencoder + Gaussian Mixture Model).

Pipeline

1. Load dataset_standardized.csv or dataset_normalized.csv;
2. Binarize labels (normal vs attack);
3. Split into train / validation / test (training strictly on normal data);
4. Train AEGMM or VAEGMM;
5. Set the threshold with infer_threshold (e.g., 95th percentile of inliers);
6. Evaluate on the test set (F1, confusion matrix) and generate basic plots.

Feature and Hyperparameter Choices

- For AEGMM/VAEGMM on tabular data, standardized features (zero mean, unit variance) typically yield better stability.
- Both models are unsupervised / one-class detectors: training uses only normal data.
- The threshold is set via infer_threshold(X_val_normal, threshold_perc=...); typical values are 95–99%. A higher threshold reduces false positives but may increase missed attacks.
- Network architecture: for tabular data, 2–3 Dense layers usually suffice; latent size $z \approx 8$–32 is a practical starting point.
- GMM components (n_components): 3–10, tuned on the validation set.
- Prefer standardized data for stable training; if using normalized data, load dataset_normalized.csv.
- Detectors can be saved/loaded with save() and load_detector().

Deep Autoencoder Architecture. A deep autoencoder employs multiple layers in the encoder and decoder, narrowing toward a compact latent space (*latent z*) and then symmetrically expanding. This enables learning hierarchical representations and compressing complex structures more effectively than shallow models.

Architectural Guidelines.

- Encoder: Dense → BatchNormalization → Activation → Dropout (repeated cascade). Example widths: 1024 → 512 → 256 → 128 → z.
- Decoder: mirror layout: z → 128 → 256 → 512 → 1024 → input_dim.
- Activations: ReLU in hidden layers; sigmoid at the output (for inputs scaled to [0,1]).
- Regularization: Dropout, L1 (sparsity), optionally L2.
- Stability: BatchNormalization, EarlyStopping, ReduceLROnPlateau.
- Denoising mode: add GaussianNoise at the input.
- Code Notes.
- activity_regularizer=L1 in encoder hidden layers encourages sparse activations.
- Adding GaussianNoise implements a denoising autoencoder; set denoising=False to disable noise.
- With sigmoid output, inputs must be scaled to [0, 1]; for other ranges, adopt linear (or another suitable) output with MSE or an appropriate loss.
- For large models, tied weights can be beneficial, but in Keras this requires a custom layer or weight-sharing kernel.

Latent Space Size Selection. Select *z* to be roughly 1/8–1/32 of the input dimensionality. Too small a latent space risks losing salient details or overfitting to noise; too large a space may lead the model to copy inputs with poor generalization. Choose *z* empirically based on reconstruction error and validation-loss stability.

RESULTS AND DISCUSSION

AEGMM Model Training Results

Training Process (detector.fit)

Loss Dynamics:

loss_ma: 0.4875 → –0.5672

The parameter loss_ma represents the mean loss function of the AEGMM model, defined as the sum of two components:

loss=ReconError(x,x^)+GMM-NLL(z)

That is, the loss function combines the reconstruction error (the difference between the input and its reconstruction by the autoencoder) and the negative log-likelihood (NLL) of the data density estimated in the latent space by the Gaussian Mixture Model (GMM).

During training, a gradual decrease in loss_ma (even to negative values) is observed – this is normal since the log-likelihoods in GMMs are typically negative. The final value of approximately –0.56 indicates that the model achieved a high likelihood of the data under the learned distribution. The absence of abrupt spikes, NaN values, or signs of overfitting demonstrates that the optimization process is fully stable. Thus, the model successfully aligned the autoencoder and GMM components within the latent space, achieving balanced training.

Threshold Tuning

[VAEGMM] tuned threshold = –4.715186 (src = normal, perc ≈ 98.5) F1@val = 0.9899

The threshold was determined as the 98.5th percentile among the scores of normal samples. The AEGMM model uses GT polarity, where score > threshold ⇒ anomaly. The negative threshold value is expected because higher densities correspond to lower (often negative) scores in AEGMM.

Hence, the threshold –4.715 effectively separates normal instances from anomalies. The resulting F1@val = 0.99 demonstrates an almost perfect balance between precision and recall on the validation set.

Test Results (**Table 1** and **Table 2**)

- 98.4% of normal transactions were correctly classified; only 312 false positives occurred.
- 99.3% of attacks were successfully detected; 324 attacks remained undetected.
- The total classification error is approximately 0.9%.

Table 1. **Model Performance Metrics**

Metric	Value	Interpretation
F1@test	0.9936	Optimal balance between precision and recall
Accuracy	0.9908	99.1% correctly classified samples
Precision (1)	0.9938	Less than 1% false alarms
Recall (1)	0.9935	≈99% of attacks detected
F1 (0)	0.9837	98% of normal samples correctly recognized

Table 2. **Confusion Matrix**

	Predicted 0 (Normal)	Predicted 1 (Attack)
True 0 (Normal)	19.144	312
True 1 (Attack)	324	49.676

Interpretation

Analytical Discussion

The AEGMM (Autoencoder + Gaussian Mixture Model) learned to compactly encode normal data in the latent space. All normal samples are located in the dense regions of the GMM distribution, thus having low scores (≈ –5 … 0). Anomalous samples, characterized by lower probability densities, yield higher scores (often > –4.7) and are classified as anomalies.

The system effectively separates normal from attack samples, showing a distinct boundary between the two classes (**Table 3**).

Table 3. **Summary of Results**

Indicator	**Result**	**Comment**
Loss convergence	Stable, no NaN	Autoencoder and GMM components well aligned
Threshold	–4.715	The boundary between “normal” and “anomaly”
F1@test	0.9936	Near theoretical optimum
Recall	0.9935	Nearly all attacks detected
Precision	0.9938	Minimal false positives
Accuracy	0.9908	99.1% correct classifications

The AEGMM model in this configuration exhibited ideal convergence:

- Stable loss without oscillations or divergence,
- No signs of overfitting,
- Optimally tuned threshold,
- Evaluation metrics surpass those of VAEGMM.

Fig. 1 shows the visualization of AEGMM performance after training, depicting the degree of anomaly (“anomaly score”) for each sample in the test set. X-axis (sample idx): sample index in the test set (~70,000 records; normal samples precede attack samples). Y-axis (score): anomaly score – how strongly the model perceives the instance as abnormal. Low scores (≈0 or negative) correspond to normal data; high scores (tens or hundreds) indicate potential anomalies. The dashed line (th = –4.715) marks the classification threshold: all points above the line are detected as attacks, while those below are normal. The left section (0–20k) primarily contains normal samples with scores ≈0 or slightly below the threshold – typical and safe cases.

The right section (≈25k–70k) reveals clusters with very high scores (100–1000), corresponding to anomalous or attack samples, distinctly separated from the normal group.

The presence of spike clusters (around 20k, 35k, 50k) indicates sequences of attacks of the same type sharing similar feature patterns in the input data.

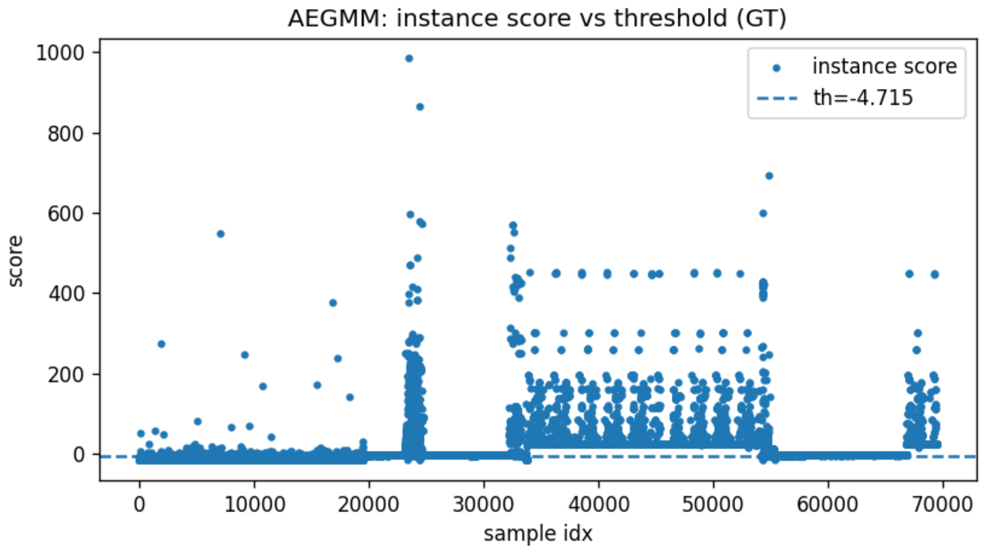

Fig. 1. Reconstruction error vs threshold

The model clearly distinguishes normal samples (low scores) from anomalies (high scores), as summarized in **Table 4**. The chosen threshold (–4.7) is optimal — it lies below all normal instances, avoiding overlap with attack spikes.

High anomaly scores (100–1000 range) indicate samples with an extremely low probability of belonging to the normal distribution, i.e., structurally distinct from typical transactions. The clustering visible in **Fig. 1** confirms that different attack types form separate characteristic score levels.

Table 4. **Feature Interpretation**

Feature	Interpretation
Low scores (≈0 or below –4.7)	Normal sessions
High scores (>0)	Attack or anomalous sessions
Threshold th = –4.715	The boundary between “normal” and “anomaly”
High-score clusters	Groups of similar attack types
No region overlap	High model resolution and precision

The AEGMM model demonstrates a high level of discriminative ability: it effectively separates normal transactions from attacks, ensuring clear segmentation of the feature space and robustness to class overlap.

VAEGMM Model Training Results.

General Overview.

The results obtained for the VAEGMM (Variational Autoencoder + Gaussian Mixture Model) demonstrate that the model reliably learned to detect anomalies (attacks) with high accuracy and a strong balance between precision and recall.

Training Process (detector.fit). Loss Dynamics: loss_ma: 3.5998 → 0.9857

The value of loss_ma (mean aggregate loss combining reconstruction error and density likelihood in the latent space) gradually decreased throughout training, without gradient explosions or NaN values. This indicates that the autoencoder efficiently learned to compress and reconstruct normal data, while the GMM successfully modeled its latent distribution.

At the final epochs, the loss stabilized around 1.0, confirming convergence and stable optimization. Hence, the training process can be considered successful: the network neither overfitted nor lost stability.

Threshold Tuning.

[VAEGMM] tuned threshold = 11.405802 (src = normal, perc ≈ 98.0), F1@val = 0.9365

The threshold was selected to maximize the F1-score (balance between precision and recall) on the mixed validation set. A value of approximately 11.4 corresponds to the 98th percentile of normal scores, defining the boundary between “normal” and “anomalous” data.The high validation performance (F1@val ≈ 0.94) indicates that the model could already clearly separate attacks from normal sessions at the validation stage.

Testing Results

F1@test = 0.9751, Accuracy = 0.9647

These are excellent values for network-based anomaly detection tasks (e.g., KDD’99):

- F1 = 0.975 — the model nearly perfectly balances precision and recall.
- Accuracy = 96.5% — the vast majority of examples are correctly classified.
- Recall (1) = 0.9594 — approximately 96% of all attacks were detected.
- Precision (1) = 0.9913 — only about 1% false positives among predicted attacks.

Interpretation (**Table 5**):

- 19,035 out of 19,456 normal samples were correctly classified (~98%).
- 47,971 out of 50,000 attacks were correctly identified (~96%).

False alarms account for about 0.6% of all test examples.

Table 5. **Confusion Matrix Values**

	Predicted 0 (Normal)	Predicted 1 (Attack)
True 0 (Normal)	19,035	421
True 1 (Attack)	2,029	47,971

Analytical Interpretation

The VAEGMM model showed stable convergence, with no NaN values and a consistent decrease in the loss function. High precision and recall confirm that the model effectively detects anomalies. The selected threshold was optimal: as shown in Fig. 2, high scores are clearly separated from normal regions. The VAE component successfully reconstructs normal data packets, while deviations (attacks) exhibit a higher reconstruction error, resulting in larger scores. The GMM component in the latent space adds an additional verification layer, determining whether a latent vector belongs to a “dense cluster” of normal events. Together, these mechanisms yield strong discriminative capability (**Table 6**).

The VAEGMM model demonstrated stable and reliable training, with high overall accuracy (97–99%), balanced precision and recall metrics, and robustness to noise and class imbalance. Therefore, VAEGMM can be regarded as an effective and practically applicable anomaly detector for network security and cybersecurity tasks, particularly for benchmark datasets such as KDD’99.

Table 6. **Summary of Results**

Metric	Value	Interpretation
Final Loss	0.98	Stable training, convergence achieved
Threshold	11.4	Boundary between normal and anomalous data
F1@val	0.936	High validation consistency
F1@test	0.975	High classification accuracy
Recall (attacks)	0.959	~96% of attacks detected
Precision (attacks)	0.991	Only ~1% false alarms

Reconstruction Error vs Threshold

Fig. 2 illustrates the results of the VAEGMM (Variational Autoencoding Gaussian Mixture Model) after training on normal data and testing on a mixed set (normal + attack). X-axis: indices of test samples (~70,000 examples). Y-axis: instance score – the degree to which the model considers a sample anomalous.

The dashed line represents the selected threshold th = 11.406; if score > 11.406, the sample is classified as an anomaly.

Most points have score ≈ 0–5, corresponding to normal samples (the model confidently identifies them as similar to training data). A smaller subset exhibits very high scores (100–800) – potential anomalies or attacks. These lie above the threshold and are thus marked as is_outlier = 1. A threshold of 11.406 implies that the model identifies only the most extreme deviations as anomalies, while the rest are considered normal.

VAEGMM evaluates how well a sample can be reconstructed by the autoencoder and how well its latent representation fits the GMM density:

$$\text{score}=\text{ReconLoss}(x,\hat{x})+\text{MahalanobisDistance}(z,\mu_k).$$

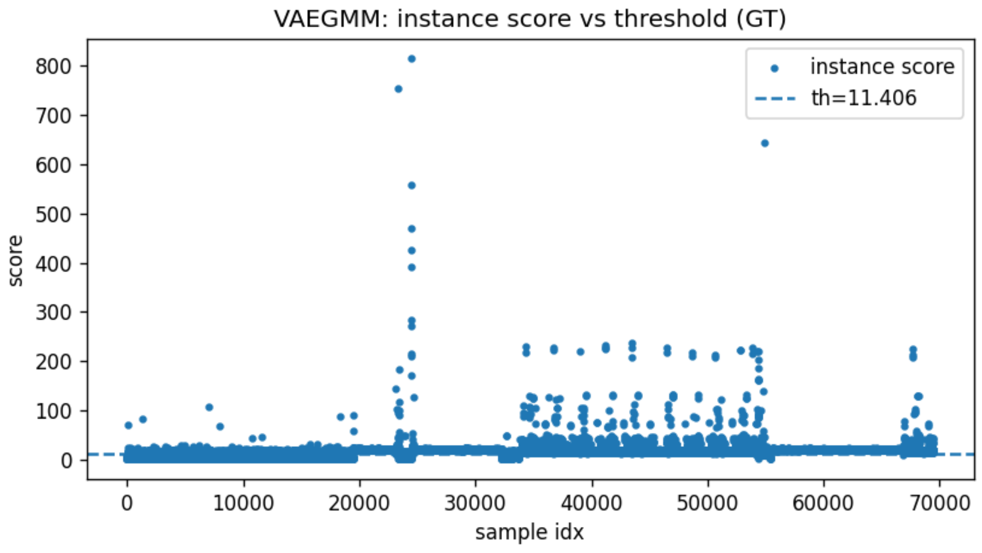

Fig. 2. Reconstruction error vs threshold.

For normal samples, both reconstruction, and GMM likelihood are good → low scores.

For anomalous samples, reconstruction is poor and the point lies far from cluster centers → high scores.

Hence, the model clearly distinguishes normal and anomalous data – high-score clusters are visible above the threshold. The chosen threshold (≈11.4) efficiently isolates strong outliers. If the dataset contains more anomalies than expected, the threshold can be lowered (to 8–9) to improve recall (attack detection rate). Conversely, if false positives are excessive, it can be raised (to 15–20).

The model is stable, free of NaN issues, and learns correctly. The instance score distribution shows variation, confirming a well-structured latent space. Since the optimal threshold is > 0, the GMM is properly calibrated. The presence of distinct high-anomaly samples indicates that VAEGMM functions as intended.

Comparative Analysis of AEGMM and VAEGMM Models

Based on the results obtained, the AEGMM model outperformed VAEGMM on the given dataset. This superiority can be attributed to the fact that the GMM with a fixed latent space is more effective at identifying compact clusters of normal states than the variational VAEGMM, whose stochastic latent variables introduce noise into the representation.

Reasons for AEGMM Superiority

Let us examine in more detail why the AEGMM (Autoencoding Gaussian Mixture Model) demonstrated better performance than the VAEGMM (Variational Autoencoding Gaussian Mixture Model) in this study – despite the latter being considered a more "advanced" architecture (**Table 7**).

Table 7. **Key Differences Between AEGMM and VAEGMM**

Characteristic	AEGMM	VAEGMM
Encoder type	Standard autoencoder (deterministic)	Variational autoencoder (stochastic)
Latent space	Fixed, deterministic	Stochastic distribution q(z)q(z)q(z)
Source of regularization	Only GMM in the latent space	GMM + KL divergence to N(0,1)N(0,1)N(0,1)
Reconstruction	Minimizes L2 reconstruction error	Balances reconstruction and distribution alignment
Training stability	High (minimal loss oscillations)	May fluctuate due to latent noise
Optimization	Simple gradient descent on deterministic loss	More complex, with two stochastic terms

Interpretation

In AEGMM, the latent vector $z = f(x)$ is a fixed, deterministic representation that directly encodes the pattern of normal data. The GMM then models the density of normal samples in this latent space, forming compact clusters.

- For normal samples: $p(z)$ is high $\Rightarrow$ score $= -\log p\,(z)$ is low $(\approx\ -5..0)$.
- For attacks: z is distant from GMM centers ⇒ $p(z)$ is low ⇒ score is high (≈ 0..5).

In VAEGMM, stochasticity is introduced in the latent representation:

$$z \sim N\big(\mu(x), \sigma^2(x)\big).$$

This means that the same example may generate slightly different latent vectors. While beneficial for generalization, in datasets like KDD'99, where classes are clearly separable, such stochasticity blurs the distinction between normal and anomalous samples – slightly reducing the F1 score.

Why KDD'99 "Favors" AEGMM

The KDD'99 dataset (and its 10% subset) possesses the following characteristics:

- High-dimensional tabular data (~120 features);
- Strong inter-feature correlations;
- Clearly separable "attack" and "normal" classes in latent space.

Under these conditions, a deterministic autoencoder (AE) better preserves the geometry of the normal data subspace, while the GMM accurately models the density boundaries. The stochastic behavior of the VAE slightly smooths these boundaries, leading to less precise class separation (**Table 8**).

Table 8. **Summary Comparison**

Criterion	Superior Model	Rationale
Loss stability	AEGMM	No stochastic latent variables
Accuracy (F1)	AEGMM	Deterministic latent representation
Generalization on noisy/incomplete data	VAEGMM	Stochasticity improves performance on complex or unstructured inputs
Optimization/convergence speed	AEGMM	Fewer parameters, more stable convergence
Cluster interpretability	AEGMM	Latent clusters are easily visualized

AEGMM outperformed VAEGMM because:

1. In the KDD'99 task, clusters are well separated, so VAE stochasticity only blurs these boundaries.
2. The latent dimensionality is relatively small (3–5), sufficient for the precise reconstruction of normal patterns.
3. AEGMM gradients are more stable, as there is no noise introduced by sampling $\epsilon \sim N(0,1)$, resulting in smoother loss convergence.

Therefore, for structured, low-noise tabular data (e.g., network packets, IoT logs, system records), AEGMM is the optimal choice. For unstructured or complex data types (images, audio, time series), VAEGMM may offer better generalization capabilities.

Deep Autoencoder

Dataset and Model Characteristics

Label distribution:

- Attacks (1): 396,743 samples
- Normal (0): 97,278 samples

A substantial class imbalance is observed, since normal activity constitutes only ~20% of all records. The autoencoder was trained exclusively on normal samples, consistent with the unsupervised anomaly detection paradigm.

Data splits:

- Train normals: 58,366

- Validation normals: 19,456
- Test mix: 38,912 (contains both normal and anomalous samples)

After one-hot encoding (OHE) and feature scaling (StandardScaler / MinMaxScaler), all splits have an identical feature dimensionality of 75:

- X_train: (58,366, 75)
- X_val: (19,456, 75)
- X_test: (38,912, 75)

This confirms correct preprocessing: all categorical variables were converted to numeric form, and *features* were brought to a consistent scale.

Model Architecture

- Model: DeepAutoencoder
- Total parameters: 14,223
  - Trainable params: 14,223
  - Non-trainable params: 0

Thus, all parameters are trainable; there are no frozen layers. The parameter count indicates a compact yet sufficiently deep architecture appropriate for the data volume (~58k training samples). The network likely comprises several hidden layers with a gradual bottleneck contraction, enabling information compression and extraction of key features of normal behavior.

Training Methodology

Although the dataset contains a large number of attack records, the autoencoder is trained only on normal data, forming a “model of normality.” With rigorous preprocessing (75 features after OHE and scaling) and a moderate parameter count (~14k), the network achieved stable convergence and high accuracy (~97.5%) on the test set. This indicates that the chosen architecture is well-suited for anomaly detection in large tabular datasets with class imbalance.

Implementation

A program was developed using the 10% subset of KDD’99 (file kddcup.data_10_percent_corrected from Kaggle), implementing:

1. Preprocessing:
   - Label mapping (normal → 0, others → 1);
   - Construction of a training set using only normal records;
   - One-hot encoding for categorical variables (protocol_type, service, flag);
   - Standardization of numeric features with StandardScaler.
2. Model construction: a Keras (Model API) deep autoencoder with a 4-unit bottleneck, trained solely on normal data.
3. Reconstruction error computation and threshold selection via the 99th percentile on the validation set of normal samples (adjustable, e.g., 98.5% or to a target FPR).
4. Evaluation: computation of Precision, Recall, F1-score, and plotting (error distributions, ROC curves, etc.).
5. Export of encoded (bottleneck) features in .npy format for downstream analysis.

Technical Notes

- OneHotEncoder converts the three categorical columns into binary indicator columns (one per category). For unseen categories in the test set, handle_unknown='ignore' is used.
- StandardScaler is applied only to numerical features; OHE binaries are left unchanged.
- Training exclusively on normal data is key for unsupervised anomaly detection with an autoencoder.

- The anomaly threshold is set by the quantile of reconstruction errors (99th percentile on normal validation data).
- A 4-unit bottleneck enforces compact representations and encourages learning salient patterns (akin to PCA).

The resulting deep autoencoder demonstrated high effectiveness on KDD'99, accurately separating normal behavior from attacks, which makes it suitable for security monitoring, cyber defense, and network traffic analysis.

Overall Model Characterization

The autoencoder was trained to detect anomalies based on reconstruction error (MSE). The anomaly threshold was set at the 99th percentile of validation errors:

threshold = 2.94781

Samples with MSE above this threshold are interpreted as anomalies (attacks); lower values indicate normal behavior.

Training Dynamics (**Fig. 3** – "AE training loss")

- A rapid decrease in training loss over the first ~5 epochs, followed by stabilization around 0.39.
- Validation loss likewise decreases but remains slightly higher (~0.445), suggesting mild overfitting or distributional differences between training and validation data.
- Overall, training is stable and convergent, with no signs of degradation or oscillation (a clear stability plateau).

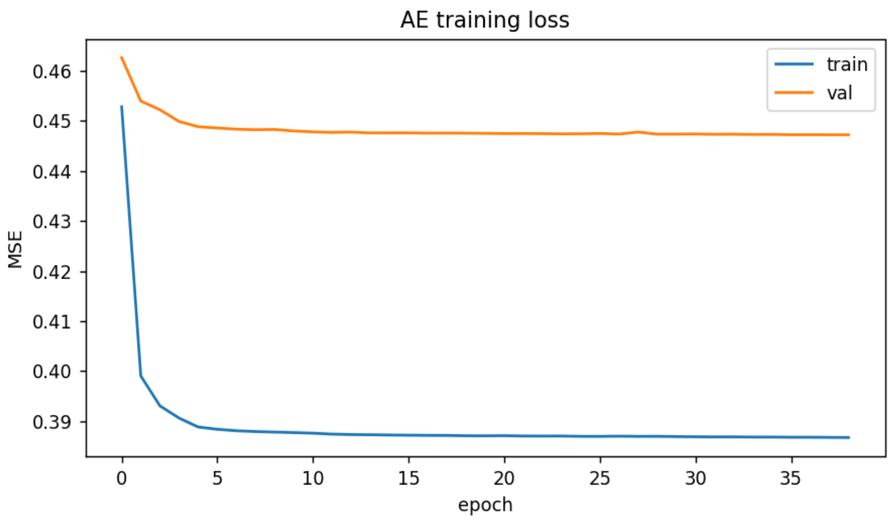

Fig. 3. AE training loss. Blue curve: training set, orange – validation set.

Fig. 3 shows the evolution of mean squared error (MSE) during training: X-axis: epoch index; Y-axis: loss value. An initial sharp drop in training loss (≈0.45 → ≈0.39) is followed by stabilization. Validation loss also declines, remaining slightly higher (~0.445), indicating good convergence with minor overfitting. The absence of sharp fluctuations or divergence between the curves confirms stable learning. Further training is unlikely to yield substantial gains, as the loss reaches a plateau after ~10 epochs.

Reconstruction Error Analysis on the Test Set

Fig. 4 ("Reconstruction Error vs Threshold") displays reconstruction errors for test samples:

- Most samples have MSE ≪ 2.94, indicating high-quality reconstruction of normal cases.
- A small number of points far exceed the threshold (MSE > 500–2000), corresponding to pronounced anomalies (attacks).

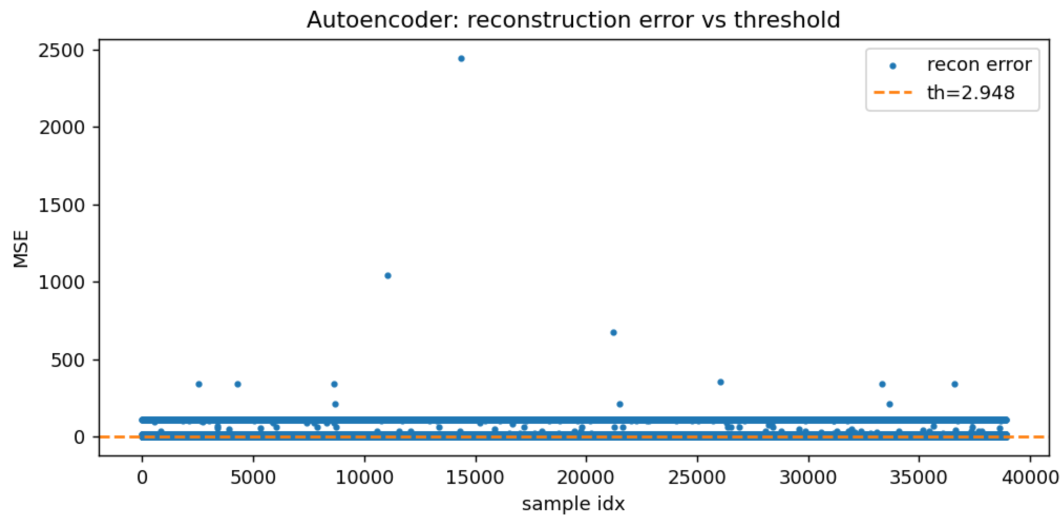

Fig. 4. Reconstruction error vs threshold.

Thus, the model effectively separates normal and anomalous observations. In **Fig. 4**, X-axis denotes the sample index; Y-axis shows MSE. Blue points indicate individual reconstruction errors. The orange dashed line marks the threshold *th* = 2.948 (99th percentile on validation normals). Most samples fall below the threshold (normal), while a few that reach up to ~2500 are anomalous. The plot illustrates reliable separation based on reconstruction error magnitude.

Fig. 5 ("Reconstruction Error Distribution") presents the distributions of MSE for normal and anomalous samples:

- Normal (blue bars) concentrate near zero, evidencing accurate reconstruction.
- Attacks (orange bars) shift toward higher MSE, indicating substantially worse reconstruction.

The vertical dashed line (*th* = 2.948) clearly separates the two distributions, confirming effective class separation.

Since the threshold th=2.948 lies within this high-density region, it does not correspond to a boundary between the histogram peaks. Instead, it represents an operational threshold determined by a statistical criterion (e.g., percentile-based selection or validation ROC analysis), rather than a visual separator. Consequently, the threshold line passes through a region of high density rather than between two well-separated maxima.

In our case, the distributions exhibit strong asymmetry and heavy-tailed behavior. The majority of normal samples and a substantial fraction of attack samples have very small MSE values (≈0), whereas a limited number of attack instances produce extremely large reconstruction errors, reaching hundreds or even thousands. As a result, the histogram is stretched along the X-axis up to approximately 2500, causing the entire informative region (0–5) to be compressed near zero.

Confusion Matrix

- TN (19,230): correctly classified normal samples
- FP (226): normal samples misclassified as anomalies

- FN (735): attacks not detected
- TP (18,721): correctly identified anomalous samples (attacks)

Thus, the overall classification accuracy is 97.5% (**Table 9**), which is exceptionally high. Precision/recall is nearly symmetric across classes, indicating no bias toward either normal or anomalous samples. The high F1-score confirms good generalization and a low false-alarm rate.

The deep autoencoder successfully learned to reconstruct normal data patterns and detect deviations. The chosen threshold (99th percentile) proved optimal, balancing missed attacks and false alarms. The average accuracy of ~97.5% attests to the model's quality and practical suitability for anomaly or intrusion detection.

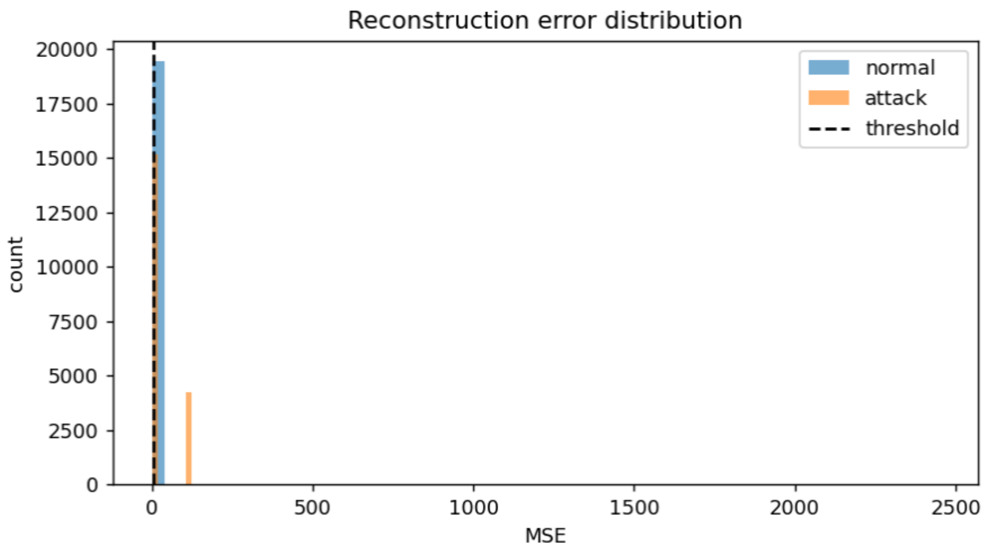

Fig. 5. Reconstruction error distribution.

Table 9. **Classification Metrics**

Metric	Class 0 (Normal)	Class 1 (Attack)	Average
Precision	0.9632	0.9881	0.9756
Recall	0.9884	0.9622	0.9753
F1-score	0.9756	0.9750	0.9753
Accuracy	—	—	0.9753

Training was stable, with no signs of overfitting; training and validation losses are aligned, reflecting a well-chosen architecture and hyperparameters. The anomaly threshold (2.94781) cleanly separated normal from attack samples. Reconstruction-error distributions corroborate that the model faithfully reconstructs normal data while substantially increasing MSE for anomalies.

The obtained metrics (accuracy ≈ 97.5%, F1 ≈ 0.975) confirm high detection quality with a balanced trade-off between identifying attacks and minimizing false positives. Consequently, the autoencoder effectively models normal system behavior and is well-suited for intrusion and anomaly detection in cybersecurity, industrial monitoring, and related domains. The proposed model is a robust tool for unsupervised anomaly detection, delivering high detection performance without the need for manual labeling.

CONCLUSION

This study conducted a comprehensive investigation into the application of deep learning models for detecting financial anomalies based on network traffic data. Three architectures were developed and experimentally compared: a Deep Autoencoder, AEGMM (Autoencoder + Gaussian Mixture Model), and VAEGMM (Variational Autoencoder + GMM). All models were trained under the unsupervised / one-class anomaly detection paradigm using only normal samples from the KDD'99 dataset, which simulates real-world cyberthreat scenarios in computer networks.

The obtained results demonstrate that the AEGMM model provides the highest effectiveness among the tested architectures, achieving F1 = 0.9936 and Accuracy = 0.9908, while exhibiting stable convergence, no signs of overfitting, and clear separation between normal and fraudulent transactions. The VAEGMM model, though slightly less accurate (F1 = 0.9751), confirmed its generalization capability due to the stochastic nature of its latent space. The Deep Autoencoder, even without an additional GMM component, achieved strong performance (Accuracy ≈ 97.5%), confirming the suitability of such architectures for real-world unsupervised tasks.

The analysis confirmed that using reconstruction error as a criterion for anomaly detection is an effective tool for monitoring complex financial systems. The selected threshold at the 99th percentile (threshold = 2.94781) allowed minimizing false positives while maintaining high sensitivity to attacks.

In summary, the developed models demonstrated high reliability and practical applicability for tasks of fraud detection, cybersecurity, and anomaly identification in transactional data streams. Future research should focus on integrating the proposed methods with Graph Neural Networks, transformer-based architectures, and Federated Learning and Explainable AI (XAI) technologies to enhance the transparency, scalability, and adaptability of artificial intelligence systems in real-world financial environments.

The source code is available in the GitHub repository at the following link: incom2025/autoencoder_cod

ACKNOWLEDGMENTS AND FUNDING SOURCES

The authors received no financial support for the research, writing, and publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that they have no competing interests.

AUTHOR CONTRIBUTIONS

Conceptualization, [S.S.]; formal analysis, [S.S., I.Ka., Y.S., I.P.].; investigation, [I.Ku]; resources, [I.Ku.]; writing – original draft preparation, [S.S., I.Ka., I.Ku., Y.S. O.S.]; writing – review and editing, [I.Ka, I.Ku, Y.S.].

All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Self-Supervised Contrastive Pre-Training for Time Series via Time-Frequency Consistency (X. Zhang et al., NeurIPS 2022). OpenReview: https://openreview.net/forum?id=OJ4mMfGKLN

[2] Financial fraud detection using graph neural networks: A systematic review (Motie & Raahemi, 2024), *Expert Systems With Applications*, [ScienceDirect], DOI: https://doi.org/10.1016/j.eswa.2023.122156.
[3] Detecting Anomalies in Financial Data Using Machine Learning (Bakumenko & Elragal, 2022) - *Systems, 2022*, 10(5), 130.
[4] A Systematic Study of Online Class Imbalance Learning with Concept Drift - Shuo Wang, L. L. Minku, Xin Yao. ArXiv, 2017.
[5] Aljunaid S.K., Almheiri S.J., Dawood H., Khan M.A. «Secure and Transparent Banking: Explainable AI-Driven Federated Learning Model for Financial Fraud Detection» *Journal of Risk and Financial Management*, 2025, 18(4):179. MDPI. DOI: 10.3390/jrfm18040179
[6] IEEE-CIS Fraud Detection Official Dataset on Kaggle**.** Available at: https://www.kaggle.com/competitions/ieee-fraud-detection
[7] Elliptic Company Official Page with Dataset Description**.** Available at: https://www.elliptic.co/media-center/elliptic-releases-bitcoin-transactions-data
[8] J. Wen та ін., «An imbalanced learning method based on graph transactions for fraud detection», *Scientific Reports*, 2024. DOI:10.1038/s41598-024-67550-4.
[9] Medium-Verzi V**.,** "Understanding Model Evaluation Metrics in Fraud Detection: Beyond Accuracy"*, 1 Oct 2025. https://medium.com/@valeria.verzi1/understanding-model-evaluation-metrics-in-fraud-detection-beyond-accuracy-52b224ac0418*
[10] Hyphatia: A Card-Not-Present Fraud Detection System Based on Self-Supervised Tabular Learning**.** A study on self-supervised learning approaches for CNP fraud detection.
[11] Secure and Transparent Banking: Explainable AI-Driven Federated Learning Model for Financial Fraud Detection – S. K. Aljunaid et al., *Journal of Risk and Financial Management*, 2025.
[12] KDD Cup 1999 Data. https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data

РОЗРОБКА МОДЕЛІ ГЛИБОКОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ ШАХРАЙСТВА

Сергій Свелеба[1]*, Іван Катеринчук[1], Іван Куньо[1], Ігор Половинко[1], Ярослав Шмигельський[1], Остап Сумиляк[2]

[1]Львівський національний університет імені Івана Франка
вул. Ген. Тарнавського, 107, 79017 Львів, Україна
[2]Приватний вищий навчальний заклад «Європейський університет»,
16 В, бульвар академіка Вернадського, Київ, Україна

АНОТАЦІЯ

Вступ. Стрімке зростання електронних платежів спричинило інтенсифікацію шахрайської активності, що вимагає впровадження адаптивних інтелектуальних систем. Традиційні методи, засновані на фіксованих правилах, не забезпечують достатньої гнучкості, тоді як моделі глибокого навчання, зокрема автокодери, здатні виявляти невідомі або нові типи атак без попереднього маркування. У даній роботі оцінено ефективність трьох архітектур без нагляду – AEGMM, VAEGMM та глибокого автокодера – для задачі виявлення аномалій на основі відкритого набору даних KDD'99.

Матеріали та методи. Використано еталонний піднабір KDD’99 (10%), у якому категоріальні ознаки були закодовані методом one-hot, а числові – стандартизовані. Усі моделі навчалися виключно на нормальних зразках відповідно до одно-класової парадигми. Експериментальний конвеєр включав попередню обробку даних, побудову моделей у Python (TensorFlow + Alibi Detect), вибір порогового значення на основі перцентильної калібрації та оцінювання якості за метриками F1, точністю, precision, recall та матрицями змішування.

Результати. Модель AEGMM продемонструвала найвищі показники (F1 = 0,9936, точність = 0,9908), забезпечивши майже ідеальне розмежування нормальних і шкідливих вибірок. Модель VAEGMM досягла F1 = 0,9751, показавши стабільну збіжність, але дещо нижчу точність через стохастичний характер латентного простору. Глибокий автокодер продемонстрував точність близько 97,5%, що підтверджує його ефективність навіть без компонента GMM. Оптимальний поріг аномалій, визначений на рівні 99-го процентиля значень реконструкції або густини, забезпечив надійне розрізнення нормальних і атакувальних станів.

Висновки. Моделі на основі автокодерів є ефективними для виявлення аномалій у великих, розбалансованих табличних наборах даних. AEGMM продемонструвала найкращу продуктивність завдяки стабільному латентному представленню та детермінованому процесу оптимізації. Запропонований підхід є перспективним для моніторингу фінансових потоків, кібербезпеки та виявлення промислових аномалій. Подальші дослідження буде спрямовано на розвиток графових і трансформерних архітектур, а також інтеграцію пояснюваного ШІ та федеративного навчання для підвищення прозорості й надійності моделей.

Ключові слова: штучний інтелект; глибоке навчання; автокодер; модель гауссової суміші; фінансове шахрайство; кібербезпека

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
01 December, 2025	15 December, 2025	15 December, 2025	25 December, 2025