Malware Classification on EMBER 2024 with 5-Fold and Temporal Evaluation: A Hybrid SAE–Top-K–LightGBM Model

[1]Sultan Tazefidan, [2]Heya Meylem, [3]Gülay Çiçek
[1,2]Department of Software Engineering, Faculty of Engineering and Architecture
Istanbul Beykent University, Sariyer, Istanbul, Türkiye
[1]sultantazefidan.1@gmail.com, [2]heyameylem96@gmail.com, [3]gulaycicek@beykent.edu.tr

I. Experimental Analysis

A. *Dataset and Methodological Summary*

1) *Dataset Description and Splitting Strategy*

The dataset used in this study is the EMBER2024 (Win32) dataset, with the EMBER 2024 GitHub Repository referenced as the data source. The EMBER 2024 dataset stands out as a globally up-to-date and comprehensive benchmark in the field of malware detection. With its high-dimensional, balanced, and rich feature structure, it enables the evaluation of machine learning and deep learning-based models under realistic conditions. The original size of the GitHub dataset is defined as approximately 1.56 million training samples and 360,000 test samples in Win32 format.

In this study, for the 5-Fold cross-validation experiment and model stability analysis, a balanced subset was constructed by selecting 3,850 benign (0) and 3,850 malicious (1) samples from each of the first 52 weeks of the dataset. Thus, a working set consisting of 400,400 samples was obtained. As shown in Figure 1, three duplicate records were identified and removed within this subset, reducing the final dataset to 400,397 samples (Benign: 200,198; Malicious: 200,199). Selecting an equal number of samples from each week kept the data chronologically balanced, provided temporal representation, and contributed significantly to reducing temporal bias by allowing the model to observe behavioral patterns across different periods. Through this approach, the model was purified from biases that may arise due to the dominance of specific weeks and thus achieved a more consistent generalization capability.

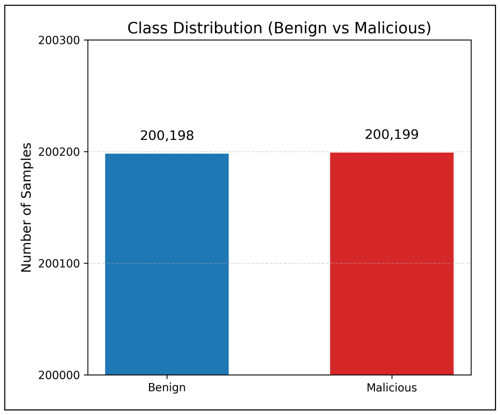

Fig. 1: Visual representation of class distribution (Benign vs. Malicious)

During the training and validation processes, the balanced subset was evaluated using the Stratified K-Fold (k = 5) method. This methodology not only reduced the risk of overfitting but also provided a more comprehensive, systematic, and methodologically robust evaluation framework compared to studies in the literature.

In addition, to support the temporal analysis, a locked final test set consisting solely of previously unseen data was prepared. This test set was constructed similarly to the strategy used in the main experiment and balanced by selecting an equal number of benign and malicious samples from the last 12 weeks of the dataset. To preserve temporal chronology, a single hold-out splitting strategy was applied instead of Stratified K-Fold at this stage, resulting in approximately 200,000 training and 200,000 test samples.

This splitting strategy aims to evaluate not only the model's performance on randomly partitioned data but also its generalization capability when encountering samples as time progresses. This comprehensive analysis is presented in detail in the following sections.

2) *Data Preprocessing and Feature Extraction*

The data preprocessing stage in this study was carried out with great rigor and a systematic approach. First, the raw .jsonl structure of the EMBER 2024 dataset was examined; label distribution, duplicate records, SHA-256 integrity verification, line-matching checks, and field structure were analyzed in detail. In the initial vectorization step, approximately 3.12 million records were identified, and after deduplication, the training set was cleaned to approximately 1.56 million samples and the test set to 359,994 samples. It was determined that 6 missing rows in the test set were incorrectly or incompletely labeled, and they were automatically removed by the filtering mechanism.

After this stage, week information was injected into the dataset to ensure chronological integrity; weeks were organized in a balanced manner from the first to the last week, and the number of samples for each week was recorded together with the corresponding log files. Thus, a "unique" 5-Fold experimental subset covering the first 52 weeks, cleaned from duplicate records and chronologically organized, was constructed. The same procedure was applied identically to the test set used in the temporal analysis.

Subsequently, since raw data were not used directly, the .jsonl files were subjected to vectorization. In this process, only the first 52 weeks of the training period were processed, and placeholder records were added for win32_train.jsonl and win32_challenge.jsonl to prevent errors. All feature vectors and mapping files produced by the thrember tool were moved to the output directory, and then the final vectorized subset was created in .parquet format by reading float32 input features and int8 labels via thrember.read_vectorized_features().

Following vectorization, comprehensive preprocessing steps were applied. First, all metadata + PE-based engineered features were converted into numerical form; then a low-variance check was applied, and 32 constant/almost constant features were removed, reducing the dimensionality to 2,536. Afterwards, a high-correlation analysis was conducted; 81 feature pairs with Pearson $|\rho| \geq 0.95$ were identified, and 31 redundant features were removed, resulting in a final feature count of 2,505.

On both the 5-fold cross-validation experiment and the temporal subset, NaN/Inf checks, duplicate scanning, memory usage monitoring, shape/dtype validation, feature index–label alignment verification, and consistency checks with original JSON files (for the temporal set) were conducted. Additionally, in all modeling steps, constant columns were re-cleaned using VarianceThreshold (threshold = 0.0) as an additional safety mechanism; after each operation, the data structure was reinforced using the bulletproof_clean() function. Since the dataset did not exhibit class imbalance, synthetic methods such as SMOTE/ADASYN were not used; instead, subsets were naturally constructed in a balanced manner, and the models were trained solely on real observations.

All fundamental data processing steps applied in this study are summarized as a unified workflow in Figure 2.

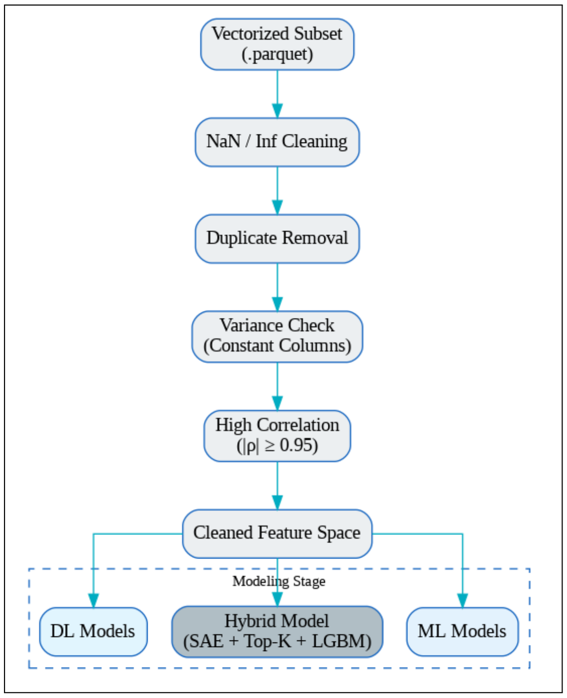
Vectorized Subset (.parquet)
NaN / Inf Cleaning
Duplicate Removal
Variance Check (Constant Columns)
High Correlation (|ρ| ≥ 0.95)
Cleaned Feature Space
Modeling Stage
DL Models
Hybrid Model (SAE + Top-K + LGBM)
ML Models

Fig. 2: Data Preprocessing and Overall Modeling Workflow

3) *Normalization*

Normalization ensures that features on different scales have a balanced influence on the model, enabling a more stable progression of the learning process. Through this operation, model parameters are updated more consistently, and gradient instabilities—particularly in deep learning architectures—are significantly reduced.

Normalization Strategy in Machine Learning Models: In machine learning–based models, the RobustScaler method was preferred due to the high concentration of outliers among the features. With its median- and IQR (Interquartile Range)-based structure, this scaler prevents outliers from adversely affecting model performance and reduces distortions arising from differences in feature scales. Especially in datasets such as EMBER, which exhibit high variance and wide value ranges, the median- and IQR-based approach of RobustScaler minimizes the dominant effect of outliers and enables more stable determination of ranking and splitting points. Consequently, tree-based models are able to make more stable decisions, achieve improved generalization capability, and perform accurate splits without being influenced by extreme values.

Normalization Strategy in Deep Learning Models: In deep learning and logistic regression models, the StandardScaler method was preferred to ensure balanced learning of network weights and stable gradient updates. This scaler transforms all features to have zero mean and unit variance, subjecting the data to z-score normalization. This transformation keeps the input values of activation functions within an ideal range (mean around 0, variance around 1), ensuring a more stable training process, particularly in neural network–based models. As a result, the risk of vanishing gradients decreases, the effectiveness of Batch Normalization and Dropout layers increases, and the model achieves faster and more stable convergence.

4) *Dimensionality Reduction Methods*

The feature distribution of the EMBER 2024 dataset exhibits a right-skewed, long-tailed, and non-Gaussian structure. While the previous version contained 2,381 features, this number increased to 2,568 in the latest release (EMBER24-v3). This increase results from the addition of new categories such as DOS header, RVH header, and PE parse warning, which contributes to more accurate detection of sophisticated malware capable of evading antivirus software, thereby enabling more reliable model training in real-world scenarios.

Although using all 2,568 features directly provides data richness, high correlation and noise may cause performance fluctuations and unnecessary complexity in some models. Therefore, in this study, both the raw-feature (OFF) structure was preserved and dimensionality reduction methods were applied to obtain more informative representations, namely PLS-DA (Partial Least Squares Discriminant Analysis) and VAE (Variational Autoencoder). Models were trained separately on the reduced representations obtained through these two methods; thus, direct performance comparisons with the original features were conducted, and the model's generalization capability and computational efficiency after dimensionality reduction were analyzed.

PLS-DA: PLS-DA is a supervised dimensionality reduction method that projects the data into a lower-dimensional subspace by maximizing the covariance between class labels and features [1]. While reducing dimensionality, it projects the samples into a space where the separation between classes becomes more pronounced. In this study, the PLS-DA approach was applied in accordance with the nature of the classification problem. During the implementation, the PLSRegression class from the sklearn library was used; since the target variable is binary (0–1), the method effectively operated according to the principles of PLS-DA. The main PLS-DA hyperparameters used in this study are presented in Table I.

TABLE I: Core PLS-DA Hyperparameters Used in the Study

Parameter	Value
Number of components ($n\_components$)	64
Scaling ($scale$)	False
Maximum iterations (max_iter)	1000
Tolerance (tol)	1×10^{-6}

VAE: The Variational Autoencoder (VAE) is a probabilistic deep learning method used for dimensionality reduction and data representation. Unlike classical autoencoders, a VAE transforms input data into a probability distribution (mean and variance) rather than a fixed latent vector. This enables the model to learn the underlying statistical structure of the data and generate new samples with similar characteristics [2]. In this study, given the structure of the dataset, the VAE model was preferred because it can effectively learn nonlinear relationships in the latent space and produce information-rich representation vectors from noisy and complex features. Using this latent representation, the performance evaluation of the hybrid model was conducted. The main β-VAE hyperparameters used in this study are summarized in Table II.

TABLE II: Core β-VAE Hyperparameters

Parameter	Value
Latent dimension	256
Encoder layers	512 $\rightarrow$ 256 (ReLU)
Regularization	Dropout(0.10), GaussianNoise(0.01)
β value (warm-up)	0.0 $\rightarrow$ 1.0 (10 epochs)
Optimizer	Adam (lr = 1e-3)
Batch size	256

5) *Threshold Optimization (Threshold Tuning with Youden's J)*

A ROC-curve–based threshold optimization was applied on the probability scores produced by the classifier. In each fold, after the model was trained solely on the training data, the ROC curve was computed on the held-out portion (the test part of the fold), and the decision threshold that maximized Youden's J statistic

$$J = \text{TPR} - \text{FPR}$$

was determined. Thus, instead of using the fixed threshold of 0.5, a dataset-specific optimal threshold was selected and consistently applied in all evaluations within the corresponding fold. In other words, in each fold, the optimal decision threshold was computed via Youden's J statistic on the ROC curve and used for the training/test evaluations of the same fold. This approach aims to reduce false negatives—critical in IDS scenarios—and to improve Recall/F1 values; in practice, it provides a more balanced error profile compared to using a fixed threshold.

Short note: For the strictest evaluation, it is recommended to select the threshold on an independent validation slice separated from the training data and apply it only on the test set. However, in this study, to preserve the consistency of stratified cross-validation, the threshold was selected on the held-out portion of each fold and the same threshold value τ^* was used in both the train and test evaluations within that fold. This approach prevented data leakage while increasing the comparability of threshold-dependent metrics.

B. *Experimental Setup*

1) *Hardware/Software Environment*

The experimental studies were conducted on a desktop computer running Windows 11 Pro 64-bit. The system is equipped with an AMD Ryzen 7 9700X processor, an NVIDIA GeForce RTX 5060 Ti (16 GB) GPU, and 32 GB DDR5 RAM, and all experiments were executed in a Python 3.12 environment using the PyCharm IDE. The primary software libraries used in the experiments are TensorFlow 2.16, scikit-learn 1.5, NumPy 1.26, and Matplotlib 3.9. All models were evaluated reproducibly under a Stratified KFold structure (n_splits = 5, random_state = 42).

In this study, all models were executed exclusively on the CPU to ensure consistent comparison; GPU acceleration was not used at any stage.

C. *Model Groups and Comparison Set*

1) *ML Models*

This group includes classical machine learning algorithms. The fundamental operating principles of each model are summarized below.

- **Gradient Boosting (GB):** An ensemble-based method that reduces the error rate by sequentially constructing weak decision trees.
- **Histogram-based Gradient Boosting (HistGB):** A faster and memory-efficient version of the GB algorithm. It reduces computational load using a histogram-based approach for large datasets.
- **CatBoost:** A boosting algorithm capable of directly handling categorical data, reducing data imbalance, and lowering the risk of overfitting.
- **ExtraTrees (Extremely Randomized Trees):** An ensemble learning method that uses randomly selected features and thresholds, providing high diversity.
- **SGDClassifier:** A fast and simple version of linear classifiers (e.g., logistic regression, SVM) optimized via stochastic gradient descent.

The machine learning algorithms selected in this study rely on complementary models capable of addressing the high-dimensional, imbalanced, and nonlinear structure of the EMBER24 dataset from different perspectives. The purpose of selecting these five algorithms together is to comprehensively analyze how different learning strategies (boosting, bagging, linear optimization) generalize on the same dataset. Thus, the balance between model complexity and accuracy was comparatively evaluated in accordance with the structure of the EMBER24 dataset.

2) *DL Models*

The deep learning group was constructed to analyze complex data patterns and representation capacity.

- **Wide Deep Model:** Learns both linear and nonlinear relationships by combining wide (memorization) and deep (generalization) components.
- **DNN (Deep Neural Network):** A multilayer feed-forward neural network structure that serves as the basic deep learning model.
- **MLP-Mixer:** MLP-Mixer is a pure MLP-based model that uses only MLP blocks—rather than convolution or attention layers—to mix features across both spatial (patch) and channel dimensions.
- **ResNet+MLP:** Captures complex representations by combining residual-learning–based convolutional layers with a multilayer perceptron (MLP).
- **gMLP-Tabular:** gMLP-Tabular is a lightweight deep network architecture that enhances the classical MLP structure with "gated" linear layers, dynamically learning interactions among features. It provides efficient generalization with a low parameter count, especially on tabular datasets.

The deep learning architectures selected in this study consist of complementary structures representing different learning strategies. While DNN represents the pure and classical MLP structure, the ResNetMLP, gMLP, and MLP-Mixer models are enhanced variants of MLP; each extends the limits of the classical MLP with a different structural innovation. The primary motivation for including the three members of the MLP family (DNN, gMLP-Tabular, and MLP-Mixer) in this study is their ability to produce fast, lightweight, and high-generalization results on tabular and high-dimensional datasets with lower parameter cost compared to convolutional or transformer-based complex architectures. Thus, the comparison of different deep architectural types was evaluated holistically based on both model complexity and representation capacity.

3) *Hyperparameter Adaptation (Common Framework for ML, DL, and Hybrid Models)*

In Table III, the core hyperparameters used in the proposed hybrid architecture are presented, while Table IV provides the fundamental hyperparameters used in the machine learning models, and the deep learning model configurations are given in Table V. These tables present the fundamental configuration settings of the models in a comparative manner. The selected parameter combinations contribute to enabling a balanced learning process and achieving high generalization performance.

Overall, the configuration values for all three model groups (ML, DL, and Hybrid) ensured the preservation of experimental integrity and enabled a fair performance comparison.

TABLE III: Core Hyperparameters of the Proposed Hybrid Model

Parameter	**Component**	**Value**
Latent Dimension	SAE (Encoder)	256
Dropout Rate	SAE (Encoder)	0.10
Epoch	SAE (Encoder)	40
Batch Size	SAE (Encoder)	512
Optimizer	SAE (Encoder)	Adam(1e−3)
Number of Features (K)	Top-K	220
Importance Ranking	Top-K	LightGBM
n_estimators	LGBM	4000
learning_rate	LGBM	0.02
num_leaves	LGBM	72
max_depth	LGBM	10
min_child_samples	LGBM	140

TABLE IV: Core Hyperparameters Used in Machine Learning Models

Model	**Parameter**	**Value**
GradientBoost	n_estimators	90
GradientBoost	learning_rate	0.1
GradientBoost	max_depth	6
HistGB	max_iter	100
HistGB	learning_rate	0.1
HistGB	l2_regularization	1.0
CatBoost	iterations	300
CatBoost	learning_rate	0.1
CatBoost	depth	8
ExtraTrees	n_estimators	500
ExtraTrees	max_features	”sqrt”
ExtraTrees	bootstrap	True
SGDClassifier	loss	log_loss
SGDClassifier	penalty	elasticnet
SGDClassifier	learning_rate	adaptive

TABLE V: Core Hyperparameters Used in Deep Learning Models

Model	**Parameter**	**Value**
WideDeep	deep_units	(512, 256, 64)
WideDeep	wide_units	1
WideDeep	dropout	0.0
WideDeep	activation	ReLU
DNN	units	(512, 256, 64)
DNN	dropout	0.2
DNN	activation	ReLU
MLP_Mixer	blocks	2
MLP_Mixer	mix	128
MLP_Mixer	hidden	256
MLP_Mixer	dropout	0.1
ResNetMLP	u1, u2, bottleneck	512, 256, 64
ResNetMLP	dropout	0.3
ResNetMLP	activation	ReLU
gMLP-Tabular (Tiny)	blocks	2
gMLP-Tabular (Tiny)	dim_ff	256
gMLP-Tabular (Tiny)	dropout	0.1
gMLP-Tabular (Tiny)	activation (gate)	Sigmoid

D. *Evaluation Metrics*

The primary metrics used for evaluating and comparing the models in this study are Accuracy, Precision, Recall, F1-Score, Weighted F1-Score (F1-W), AUC-ROC, and Specificity.

All models were evaluated using the 5-fold Stratified K-Fold cross-validation method. This approach ensures that each model demonstrates consistent performance across different subsets of the dataset, allowing its generalization capability to be measured in a statistically reliable manner. Moreover, by averaging the results obtained from each fold, the variance effect is reduced, and the reproducibility of the models is strengthened. A simplified pipeline illustrating the overall flow of this evaluation process is presented in Figure 3.

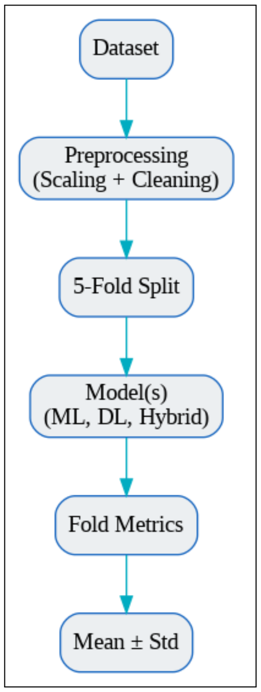

Fig. 3: The 5-fold cross-validation pipeline applied to all models (ML, DL, and Hybrid)

E. Details of the Hybrid Model

The hybrid model proposed in this study consists of a multi-stage architecture composed of a deep learning–based SAE structure, a gain-based feature selection approach (Top-K), and a tree-based gradient-boosted LightGBM classifier. As in all models, essential preprocessing steps—such as removing outliers, cleaning NaN/Inf values, and applying variance-threshold checks—were carried out to ensure data consistency and integrity prior to analysis. Additionally, *StandardScaler*, fitted only on the training subset of each fold, was used for data standardization, thereby reducing the potential impact of scale differences among variables on model performance.

1) SAE

The SAE is a deep learning architecture in which multiple autoencoders are sequentially stacked to learn hierarchical latent representations of the data [3]. The SAE architecture used in this study is a semi-symmetric deep learning structure designed to transform the high-dimensional feature space into more compact and meaningful representations. The encoder consists of fully connected (Dense) layers with 512, 256, and 256 neurons, respectively, with ReLU activation applied at each layer. To increase training stability and reduce internal covariate shift, a Batch Normalization layer is added after every dense layer. Additionally, a Dropout rate of 0.10 is applied to reduce the risk of overfitting.

The 256-dimensional latent representation obtained at the output layer of the encoder provides a noise-reduced and information-rich summary of the data. In the decoder, this latent vector is symmetrically passed through a 256-neuron layer and reconstructed using a linear activation layer with the original dimensionality at the output. The model is trained using the Adam optimization algorithm (learning rate = 1e-3) and the mean squared error (MSE) loss function. This structure enables the capture of complex patterns in nonlinear forms and allows more meaningful feature representations to be learned.

2) Top-K Feature Selection

Top-K feature selection is a method in which the best K features contributing most to model performance are selected based on statistical importance or weight scores. In this way, unnecessary or low-impact variables are eliminated, resulting in faster, more generalizable, and simpler models [4]. In this study, Top-K–based feature selection was applied to reduce model complexity and retain only the most informative variables. First, a "probe" LGBM model was trained solely on the training data, and the feature_importances_ values produced by the model were used to quantitatively measure each feature's contribution to classification performance. The obtained importance scores were sorted in descending order, and the top K = 220 features with the highest contribution were selected.

During this process, an error-prevention mechanism was added to handle cases in which no importance scores might be available, ensuring that all features could be safely evaluated. The selected Top-K features were extracted from the X_train_final32 and X_test_final32 matrices via indexing (Xtr_topK, Xte_topK) and then horizontally concatenated (np.hstack) with the latent representations obtained from the SAE. Thus, the final feature space was constructed to include both deep learning representations and the features providing the highest information gain. This approach increased the computational efficiency and generalization capability of the model by eliminating noisy or low-impact features.

3) LGBM Classifier

The LGBM classifier is a machine learning algorithm that trains decision trees sequentially and produces fast, high-accuracy predictions [5]. It is highly effective in terms of memory efficiency and processing speed, particularly when the dataset is large and the number of features is high. In this study, the classification stage of the Hybrid architecture was carried out using the LGBM algorithm. LGBM is a gradient-boosted method built on decision trees and provides both high accuracy and computational efficiency on large-scale datasets. The model was trained on an expanded feature space formed by combining the latent representations obtained from the SAE with the features selected via the Top-K method. During training, optimally tuned parameters were used to maintain a balance between model complexity and generalization capability. These parameter settings enabled the model to avoid overfitting and establish a more robust decision boundary. During the training process, an early stopping mechanism was activated, and evaluation was performed based on the AUC and binary_logloss metrics.

F. Deep Analysis of the Proposed Hybrid Architecture

1) Ablation Analysis

Table VI presents the comparative performance results demonstrating the effect of SAE integration—one of the core components of the proposed hybrid architecture—on the LGBM, LGBM+SAE, and LGBM+Top-K models. This table quantitatively reveals the contribution of these components to the model's accuracy level and computational time, clearly illustrating the role of the hybrid structure in performance optimization.

TABLE VI: Comparative performance results of LGBM, LGBM+SAE, and LGBM+Top-K models

Model	Metric	Result (Test Mean ± Std)
LGBM	Accuracy	0.9788 ± 0.0004
	Precision	0.9831 ± 0.0002
	Recall	0.9744 ± 0.0008
	Specificity	0.9832 ± 0.0002
	F1	0.9787 ± 0.0004
	F1-W	0.9788 ± 0.0004
	AUC-ROC	0.9977 ± 0.0001
	Train Time (s)	≈ 500
LGBM+SAE	Accuracy	0.9678 ± 0.0004
	Precision	0.9717 ± 0.0008
	Recall	0.9637 ± 0.0009
	Specificity	0.9720 ± 0.0009
	F1	0.9677 ± 0.0004
	F1-W	0.9678 ± 0.0004
	AUC-ROC	0.9950 ± 0.0001
	Train Time (s)	≈ 313
LGBM+Top-K	Accuracy	0.9780 ± 0.0005
	Precision	0.9823 ± 0.0017
	Recall	0.9736 ± 0.0025
	Specificity	0.9825 ± 0.0017
	F1	0.9779 ± 0.0005
	F1-W	0.9780 ± 0.0005
	AUC-ROC	0.9974 ± 0.0001
	Train Time (s)	≈ 210

All three models achieved high accuracy rates; however, slight decreases in some metrics were observed with the addition of the SAE. This can be explained by the fact that SAE may lead to limited information loss during feature representation, while LGBM already possesses strong generalization capability with raw features. In contrast, the integration of SAE reduced the computation time from 510 seconds to 313 seconds, yielding approximately a 38% speed improvement. This result demonstrates that SAE is an efficient component that reduces computational cost through dimensionality reduction while causing only marginal loss in accuracy.

Since LGBM can directly capture complex relationships due to its decision-tree-based structure, adding SAE does not always provide an additional accuracy gain. However, SAE increases the overall efficiency of the model, especially on large or high-dimensional datasets, reduces overfitting, and significantly shortens computation time. Thus, in exchange for a small decrease in accuracy, it provides meaningful computational efficiency and methodological contribution.

Furthermore, the LGBM + Top-K model produced nearly identical accuracy and AUC-ROC values compared to the baseline LGBM; the differences between the metrics remained statistically insignificant (e.g., 97.88 $\rightarrow$ 97.80 ACC, 99.77 $\rightarrow$ 99.74 AUC). However, the Top-K selection reduced the training time from 510 seconds to approximately 210 seconds, providing a substantial speed improvement. This indicates that Top-K successfully eliminates redundant features, making the model lighter and more optimized while still preserving its generalization capability. The fact that all three models achieve AUC-ROC values above 99.5% demonstrates that the discriminative power between classes remains intact.

The high accuracy, AUC-ROC, and consistent F1 values obtained are not solely the result of the architectural design but also reflect the meticulous execution of the hyperparameter optimization process. The careful tuning of critical parameters in all three models ensured stable and balanced performance. Consequently, the stable success of the model was achieved not through automatic optimization, but through deliberate hyperparameter design—demonstrating that the experimental success is driven by a design-oriented, engineering-based contribution.

2) *Effect of Feature Selection on the Final Hybrid Model*

Table VII clearly demonstrates the impact of Top-K feature selection on the performance of the hybrid model. With the application of Top-K, the OFF data representation achieved 98.30% accuracy, 98.53% precision, 98.07% recall, 98.54% specificity, 98.30% F1-score, 98.30% weighted F1-score, and 99.83% AUC-ROC. These results confirm that the model possesses both high predictive success and strong learning capacity capable of effectively representing complex data distributions. In particular, the Top-K component strengthens the model's robustness to noise by selecting only the most informative parts of the deep representations generated by the SAE, thereby enhancing its generalization ability. As a result, the hybrid model has been able to effectively capture both statistical and structural patterns within large-scale network traffic data.

TABLE VII: Hybrid Model 5-Fold Cross-Validation Results (OFF)

Metric	Result (Test Mean ± Std)
Accuracy	0.983 ± 0.0004
Precision	0.9853 ± 0.0006
Recall	0.9807 ± 0.0008
Specificity	0.9854 ± 0.0006
F1-Score	0.983 ± 0.0004
F1-Weght	0.983 ± 0.0004
AUC-ROC	0.9983 ± 0.0001

3) *Stability and Reliable Classification Analysis for the Hybrid Architecture*

The low standard deviation values obtained in the K-Fold results indicate that the models exhibit consistent performance across different training partitions. The exceptionally low variance observed particularly in the hybrid model confirms that it achieves similar accuracy levels across all folds and demonstrates strong robustness against variability caused by data splitting. This finding strongly supports that the hybrid architecture not only achieves high performance but also provides stable and reliable classification capability.

The hybrid model also demonstrates strong performance in error metrics such as FPR (False Positive Rate), FNR (False Negative Rate), NPV (Negative Predictive Value), and FDR (False Discovery Rate), which are often omitted in many studies in the literature but are critically important for intrusion detection systems (see Table VIII). The FPR and FNR values of the hybrid model, at 1.46% and 1.93%, respectively, show that both the false alarm rate and the likelihood of missed attacks are kept remarkably low in the attack class. These findings indicate that the hybrid architecture can accurately distinguish between attack and normal samples in network traffic and that it provides reliable classification performance by consistently minimizing error rates.

TABLE VIII: Additional Error Rates and Accuracy Metrics of the Hybrid Model (OFF–Raw Data)

Metric	Description	Value (%)
FPR	False Positive Rate	1.465
FNR	False Negative Rate	1.935
NPV	Negative Predictive Value	98.07
FDR	False Discovery Rate	1.466

4) *Model Size and Computational Performance Metrics of the Hybrid Architecture*

In this study, the evaluation of the hybrid architecture developed on a complex and high-dimensional dataset considered not only accuracy metrics but also performance measures that directly affect real-time system efficiency. Table IX presents the computational performance of the hybrid model, including model size, inference time, latency, throughput, and training time. All values were measured in `float32` format, and all computations were carried out on the CPU.

TABLE IX: Computational Performance Metrics of the Hybrid Model on Raw Data

Metric	Value	Description
Model Size	28.67 MB ($\approx$ 1.48M parameters)	Total model size
Inference Time	1.236 s (n = 80 079)	Total inference time
Latency	0.015 ms/sample	Average latency
Throughput	64 780 samples/s	Processing speed
Training Time	493.039 s	Average training time

As shown in the table, the hybrid model's average latency is 0.015 ms/sample, the inference time is 1.236 s (n = 80,079), and the processing speed (throughput) is calculated as 64,780 samples/s. The model size is 28.67 MB, which corresponds to approximately 1.48 million trainable parameters. This result demonstrates that the proposed hybrid architecture offers a lightweight and deployment-friendly structure.

The obtained computational performance indicates that the hybrid architecture operates highly efficiently for real-time attack detection; with its low latency, high throughput, and compact model size, it provides seamless and scalable analysis capability in real-time applications. Although the training time averages 493.039 seconds per fold, the balanced and high performance produced by the architecture—despite its complexity—renders this duration an acceptable computational cost. Therefore, the proposed hybrid model distinguishes itself not only in terms of accuracy but also through its computational efficiency, practical applicability, and real-time usage potential, setting it apart clearly from similar approaches in the literature.

5) *Classification Performance of the Hybrid Model on the OFF Representation (Confusion Matrix)*

The hybrid model's confusion matrix presented in Figure 4 demonstrates highly balanced and strong classification performance on the OFF data representation. According to the confusion matrix, the model correctly classified 197,278 benign samples as Benign and 196,326 malicious samples as Malicious. In contrast, only 2,920 benign samples were incorrectly labeled as attacks (False Positive), while 3,873 malicious samples were classified as benign (False Negative). This distribution indicates that the model maintains both false alarm rates and missed attack rates at remarkably low levels. The high values observed in the True Positive and True Negative cells clearly show that the hybrid architecture possesses strong generalization capability in distinguishing both benign and malicious traffic. In conclusion, the confusion matrix produced by the hybrid model on the OFF representation demonstrates that the model can reliably differentiate in practical IDS scenarios, keeps false positive/negative rates to a minimum, and operates with high accuracy.

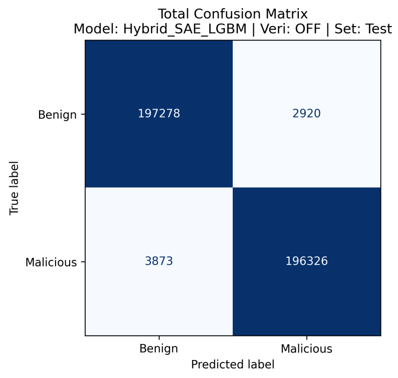

Fig. 4: Confusion Matrix Result of the Hybrid Model

6) *Statistical Stability Analysis for the Hybrid Model (Bootstrap 95% CI)*

In this section, the statistical stability of the final Hybrid model was evaluated using the bootstrap method. Within the 5-Fold cross-validation protocol, 95% confidence intervals for each performance metric were computed based on 1000 repeated bootstrap samplings applied to the test set of each fold. The aggregated results across all folds are reported in Table X. According to the bootstrap analysis, the confidence intervals for all metrics are extremely narrow (typically within the ±0.001–0.0015 range), confirming that the model's results are not influenced by random sampling variance and that it exhibits highly stable performance. In particular, the AUC-ROC value (0.9983; CI: [0.9979, 0.9986]) indicates that the model's discriminative power is at a near-perfect level. Overall, the proposed SAE+LGBM hybrid architecture demonstrated highly consistent and stable performance in terms of accuracy, precision, and general reliability.

TABLE X: Temporal Test Performance of the Hybrid Model and 95% Confidence Intervals (Bootstrap, $B = 1000$)

Metric	**Mean**	**CI_low**	**CI_high**
Accuracy	0.9830	0.9817	0.9842
Precision	0.9853	0.9840	0.9866
Recall	0.9807	0.9789	0.9818
Specificity	0.9854	0.9842	0.9869
F1 Score	0.9830	0.9815	0.9841
F1 Weighted	0.9830	0.9817	0.9843
AUC-ROC	0.9983	0.9979	0.9986

G. *Comparison of the Proposed Hybrid Model with the Literature Hybrid Model*

The hybrid model proposed in this study consists of a sequential combination of SAE-based deep representation learning, Top-K feature selection, and the LightGBM classifier. While SAE reduces the data into a more meaningful latent space, the Top-K method simplifies the model by selecting the features with the highest information gain and increases its efficiency. In the final stage, LightGBM achieves high accuracy, low latency, and strong generalization performance by utilizing these deep + selected representations.

Table XI presents a comparison between the BVR-SFO-AEDL hybrid model in the literature and the SAE + Top-K + LGBM hybrid model proposed in this study, based on core malware classification metrics. The BVR-SFO-AEDL model in the literature is a deep hybrid architecture consisting of Autoencoder–1D-CNN–BiLSTM components and has been reported to achieve a high accuracy level of 96.47% in Windows-based malware detection [6]. In contrast, the architecture proposed in this study offers a more balanced, highly generalizable, and computationally efficient structure by combining the representation learning capability of the SAE layer, the optimization effect of Top-K feature selection, and the strong decision mechanism of LightGBM.

The results demonstrate that the proposed hybrid model not only achieves a high accuracy value (98.30%) but also exhibits a distinct advantage over the literature hybrid model in terms of the recall metric (98.07%), which directly reflects attack detection sensitivity. Therefore, the proposed hybrid model provides a more stable, reliable, and practically applicable solution in terms of both overall classification performance and attack detection capability.

TABLE XI: Comparison with the Literature Hybrid Model Across Core Metrics

Model	**Accuracy**	**Recall**	**Specificity**	**AUC-ROC**
Proposed Hybrid	0.9830	0.9807	0.9854	0.9983
Literature Hybrid [6]	0.9647	0.9648	0.9632	——

Furthermore, the AUC-ROC metric is not explicitly reported in the literature hybrid model BVR-SFO-AEDL. However, in malware classification problems, AUC-ROC is critically important for measuring the model's overall discriminative power and its performance in distinguishing between classes. This omission prevents the referenced literature study from being fully comparable in terms of classification performance. In contrast, in this study, the AUC-ROC metric was explicitly calculated, and the model achieved an AUC-ROC value of 99.83%. Therefore, the proposed hybrid model distinguishes itself from existing hybrid approaches in the literature by providing transparent, complete, and comparable reporting of performance metrics.

1) *Analysis of Class-Based ROC Curves and AUC Values of the Hybrid Model*

Figure 5 presents the ROC curves obtained from the overall cross-validation (5-fold CV) results of the proposed hybrid model. As shown in Table XII, the model achieved an AUC value of 0.9983 for both the Benign (0) and Malicious (1) classes, with an average AUC score of 0.99827 (±0.00007). These results demonstrate that the proposed hybrid architecture possesses exceptionally high discriminative capability between the classes, virtually eliminating misclassifications and exhibiting outstanding stability in generalization performance.

Therefore, the hybrid model not only achieves high accuracy but also provides top-level reliability for real-time attack detection through its statistically near-perfect discriminative performance.

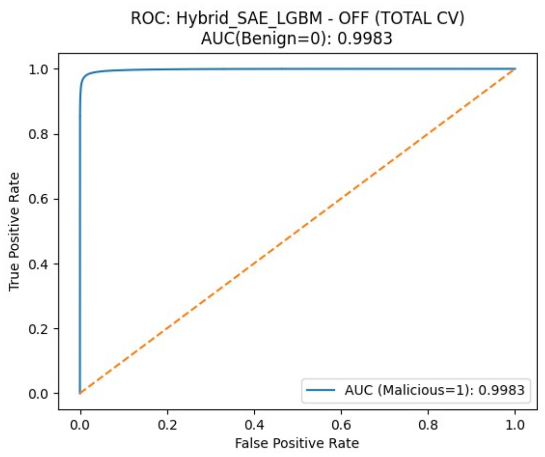

Fig. 5: Overall ROC curve of the proposed hybrid model obtained with 5-fold cross-validation on the OFF data representation

TABLE XII: Class-wise AUC performance results of the Hybrid Model

Class	AUC Value	Mean AUC (± Std)
Benign (0)	0.998271	0.998272 (± 0.000065)
Malicious (1)	0.998271	

H. *Comparative Classification Results and Statistical Analysis*

1) *ML Models: Comparative Performance Analysis*

The comparative results presented in Table XIII reveal the overall classification performance of different machine learning algorithms on the test set. The findings indicate that tree-based methods (CatBoost, ExtraTrees, GradientBoost, HistGB) exhibit significantly higher performance compared to the linear-based SGDClassifier.

The CatBoost model stands out with 97.24% accuracy and 99.61% AUC-ROC, showing stable classification behavior due to its balanced precision–recall values (97.7%–96.7%). ExtraTrees achieved the highest accuracy (97.54%) and AUC-ROC (99.72%) among all models, making it statistically the best-performing model. This result demonstrates that training a large number of decision trees on randomly selected feature subsets strengthens the model's generalization capability.

The GradientBoost model showed relatively lower performance with 95.27% accuracy and 99.10% AUC-ROC; this difference is likely attributable to hyperparameter effects, particularly learning rate and data imbalance. Similarly, the HistGB algorithm produced low-variance and stable results with 96.64% accuracy and 99.49% AUC-ROC.

To examine the linear separability of the dataset, the SGDClassifier was also tested. As expected, due to its reliance on linear decision boundaries, it yielded low performance (60.47% accuracy, 60.47% AUC-ROC). This finding empirically confirms that the EMBER dataset contains complex and nonlinear relationships. In contrast, methods capable of capturing nonlinear patterns — such as KNN or tree-based models — demonstrated much higher accuracy and generalization performance. However, the KNN model has practical limitations in large-scale scenarios due to its high computational cost and memory requirements.

TABLE XIII: Test Performance Results of Machine Learning Models on Raw Data (Mean ± Std)

Model	Metric	Result (Test Mean ± Std)
CatBoost	Accuracy	0.9724 ± 0.0004
	Precision	0.9772 ± 0.0009
	Recall	0.9674 ± 0.0011
	Specificity	0.9774 ± 0.0009
	F1	0.9723 ± 0.0004
	F1-W	0.9724 ± 0.0004
	AUC-ROC	0.9961 ± 0.0001
ExtraTrees	Accuracy	0.9754 ± 0.0007
	Precision	0.9826 ± 0.0007
	Recall	0.9681 ± 0.001
	Specificity	0.9828 ± 0.0006
	F1	0.9753 ± 0.0007
	F1-W	0.9754 ± 0.0007
	AUC-ROC	0.9972 ± 0.0001
GradientBoost	Accuracy	0.9527 ± 0.0007
	Precision	0.9561 ± 0.0009
	Recall	0.9490 ± 0.0014
	Specificity	0.9564 ± 0.0009
	F1	0.9525 ± 0.0007
	F1-W	0.9527 ± 0.0007
	AUC-ROC	0.9910 ± 0.0001
HistGB	Accuracy	0.9664 ± 0.0008
	Precision	0.9717 ± 0.0005
	Recall	0.9607 ± 0.0016
	Specificity	0.9721 ± 0.0007
	F1	0.9662 ± 0.0008
	F1-W	0.9664 ± 0.0008
	AUC-ROC	0.9949 ± 0.0002
SGDClassifier	Accuracy	0.6047 ± 0.0176
	Precision	0.5927 ± 0.0136
	Recall	0.6682 ± 0.0506
	Specificity	0.5411 ± 0.0317
	F1	0.6276 ± 0.0266
	F1-W	0.6026 ± 0.0165
	AUC-ROC	0.6047 ± 0.0176

SIn conclusion, the findings strongly support the rationale for preferring tree-based and deep learning–based models. In particular, the ability of methods such as CatBoost, ExtraTrees, HistGB, and GradientBoost to model nonlinear feature interactions, along with their high AUC-ROC scores, demonstrates that these approaches provide strong generalization on complex cybersecurity data.

However, the proposed hybrid architecture surpasses all ML-based approaches on the same dataset in terms of both accuracy and AUC metrics by integrating the representation power of deep learning with the decision-tree–based discriminative capacity of LGBM. This outcome is achieved not only through the hybrid model's ability to capture nonlinear relationships among features, but also through SAE's enhancement of information density via deep representations and LGBM's optimization power to effectively separate these representations.

From a training-cost perspective, tree-based ML models were observed to train within approximately 290–580 seconds; for a dataset consisting of 400,397 samples, these durations are expected and normal under fully CPU-based (non-GPU) execution. Furthermore, both these models—and the LGBM module that remains active at inference time in the hybrid architecture—exhibit low inference latency. Therefore, the hybrid architecture offers not only the highest accuracy and AUC performance but also the most suitable structure for real-time usage scenarios in terms of cost–performance balance.

2) *DL Models: Comparative Performance Analysis*

In the experiments conducted on the OFF data type, various deep learning architectures—including DNN, MLP-Mixer, ResNet-MLP, WideDeep, and gMLP—were tested, and the results are presented in Table XIV. The EMBER24 dataset has a high-dimensional, long-tailed, sparse, and nonlinear feature distribution, making it a challenging structure that tests the generalization capacity of such models. Despite this, thanks to the developed code infrastructure and carefully selected hyperparameter settings, remarkably high results were obtained in this type of tabular data problem, where the literature typically reports lower accuracy and AUC values.

TABLE XIV: Test Performance Results of Deep Learning Models on Raw Data (Mean ± Std)

Model	**Metric**	**Result (Test Mean ± Std)**
Wide&Deep	Accuracy	0.9706 ± 0.0013
	Precision	0.9726 ± 0.0028
	Recall	0.9684 ± 0.0021
	Specificity	0.9727 ± 0.0029
	F1	0.9705 ± 0.0012
	F1-W	0.9706 ± 0.0013
	AUC-ROC	0.9944 ± 0.0003
DNN	Accuracy	0.9707 ± 0.0011
	Precision	0.9742 ± 0.0012
	Recall	0.9669 ± 0.0024
	Specificity	0.9744 ± 0.0013
	F1	0.9706 ± 0.0012
	F1-W	0.9707 ± 0.0011
	AUC-ROC	0.9945 ± 0.0003
MLP-Mixer	Accuracy	0.9700 ± 0.0007
	Precision	0.9740 ± 0.0022
	Recall	0.9657 ± 0.0033
	Specificity	0.9742 ± 0.0023
	F1	0.9698 ± 0.0008
	F1-W	0.9700 ± 0.0007
	AUC-ROC	0.9949 ± 0.0003
ResNetMLP	Accuracy	0.9708 ± 0.0006
	Precision	0.9755 ± 0.0023
	Recall	0.9658 ± 0.0017
	Specificity	0.9758 ± 0.0024
	F1	0.9707 ± 0.0005
	F1-W	0.9708 ± 0.0006
	AUC-ROC	0.9949 ± 0.0002
gMLP	Accuracy	0.9719 ± 0.0006
	Precision	0.9756 ± 0.0033
	Recall	0.9681 ± 0.0039
	Specificity	0.9757 ± 0.0035
	F1	0.9718 ± 0.0007
	F1-W	0.9719 ± 0.0006
	AUC-ROC	0.9950 ± 0.0003

The test accuracies of the DL models ranged approximately between 97.0–97.2%, while their AUC values were in the 0.994–0.995 ((±0.0003)) range. This level of performance is not solely due to the model architecture, but is largely the result of optimized data preprocessing, scaling, regularization, and early stopping strategies. The success originates from the holistic design of the pipeline rather than from the specific type of model.

Although deep learning–based models appear faster in terms of training time at first glance, they completed training within the 200–435 s range on CPU; models with multilayer structures—such as MLP-Mixer—reached the upper end of this range. Furthermore, these models exhibit significantly higher inference time compared to the hybrid model due to their deep, matrix-multiplication–heavy structures. In contrast, despite the hybrid architecture requiring approximately 493 s of training time, it produces substantially lower latency during inference since only the LGBM module is used at this stage. This is a major advantage for real-time IDS or malware detection, where training is performed once, whereas inference runs continuously. Consequently, the hybrid model not only achieved superior training and test performance (AUC = 0.9983 ± 0.0001), but also provided operational efficiency well-suited for real-time applications.

Moreover, these results reveal a meaningful difference when compared to the proposed hybrid architecture. On the same dataset, the hybrid model achieved 98.3% accuracy, 99.83% AUC, and 98.07% recall, outperforming the DL-based approaches. In DL models, recall values remained within the 96.57–96.81% range. This gap arises from the hybrid architecture's ability to leverage both the deep representations produced by the SAE and the decision-tree–based learning capacity of LGBM. As a result, although DL-based models can produce strong outcomes on tabular data when well optimized, the hybrid architecture surpassed this performance with approximately +1.2% accuracy, +0.003 AUC, and +1.2–1.5 percentage points improvement in recall, establishing superiority in overall performance.

Figure 6 presents a comparative overview of the core metric results for ML, DL, and hybrid models in malware classification. This visualization displays the accuracy, recall, and AUC-ROC performances of the models together, clearly revealing the overall classification effectiveness of each approach.

Overall, the proposed hybrid model demonstrates higher accuracy, stability, and generalization capability compared to both classical machine learning and deep learning–based approaches. Although machine learning models (e.g., CatBoost, ExtraTrees) produced high accuracy and AUC values, their capacity to represent complex feature interactions at a deep level remains more limited compared to the hybrid architecture. Deep learning models, despite their strong representational power, performed below the hybrid structure due to computational cost.

Thus, the proposed model meaningfully differentiates itself from existing hybrid approaches in the literature through its high accuracy, low error rate, and practical applicability on complex cybersecurity data.

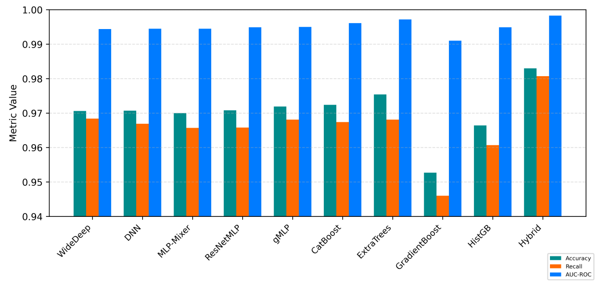

Fig. 6: Comparative Presentation of Accuracy, Recall, and AUC-ROC Metrics for DL, ML, and Hybrid Models

3) *Additional Analysis: Effect of the Scaler on Deep Learning Models*

In this study, the impact of scaler selection on DL performance was examined, and the results are presented in Table XV. The results obtained with StandardScaler significantly decreased when RobustScaler was used. The performance drop is particularly evident in the DNN, ResNetMLP, and WideDeep models, while it remains more limited in gMLP and MLP-Mixer. A possible reason for this is that the influence of outliers is already reduced after data cleaning, and the mean–standard deviation–based normalization of StandardScaler aligns gradient scales and their interaction with Batch/LayerNorm more effectively during DL training. In contrast, RobustScaler (median–IQR) may excessively compress certain features, reducing the signal-to-noise ratio and causing activations to saturate.

Since fold standard deviations remained low, the observed differences stem not from randomness but from sensitivity to the choice of scaler.

Recommendation: For this dataset/setup, keep StandardScaler as the default choice on the DL side. In cases involving severe outlier spikes, apply winsorization/clipping before StandardScaler or, if necessary, add transformations such as Quantile or Yeo–Johnson. If RobustScaler is to be used, retune hyperparameters such as learning rate and weight decay.

TABLE XV: Test Performance Results of Deep Learning Models on Raw Data (Mean ± Std) — RobustScaler

Model	Metric	Result (Test Mean ± Std)
DNN	Accuracy	0.6455 ± 0.0024
	Precision	0.5865 ± 0.0018
	Recall	0.9873 ± 0.0016
	Specificity	0.3038 ± 0.0058
	F1	0.7358 ± 0.0012
	F1-W	0.5987 ± 0.0038
	AUC-ROC	0.7407 ± 0.0112
MLP-Mixer	Accuracy	0.9159 ± 0.0013
	Precision	0.9368 ± 0.0082
	Recall	0.8920 ± 0.0092
	Specificity	0.9397 ± 0.0091
	F1	0.9138 ± 0.0016
	F1-W	0.9158 ± 0.0013
	AUC-ROC	0.9726 ± 0.0007
ResNetMLP	Accuracy	0.6511 ± 0.0046
	Precision	0.7427 ± 0.2107
	Recall	0.7176 ± 0.0364
	Specificity	0.5846 ± 0.3714
	F1	0.6342 ± 0.1403
	F1-W	0.6068 ± 0.0062
	AUC-ROC	0.7483 ± 0.0103
Wide&Deep	Accuracy	0.6746 ± 0.0832
	Precision	0.6601 ± 0.0918
	Recall	0.7463 ± 0.0717
	Specificity	0.6030 ± 0.1491
	F1	0.6980 ± 0.0688
	F1-W	0.6704 ± 0.0873
	AUC-ROC	0.7354 ± 0.0731
gMLP	Accuracy	0.9216 ± 0.0014
	Precision	0.9355 ± 0.0055
	Recall	0.9057 ± 0.0069
	Specificity	0.9375 ± 0.0061
	F1	0.9203 ± 0.0016
	F1-W	0.9215 ± 0.0014
	AUC-ROC	0.9750 ± 0.0007

4) *Evaluation of Overfitting and Underfitting Conditions*

During the model development process, the possibilities of overfitting and underfitting were carefully evaluated, yet no such issues were observed in any of the models. The fact that the training and test set results remained very close to each other clearly demonstrated that the models possess high generalization capability. In particular, to mitigate the risk of underfitting, model complexity and learning parameters were balanced appropriately, ensuring that the models adequately learned the patterns present in the data.

Additionally, the use of a 5-fold cross-validation strategy ensured that each model exhibited consistent performance across different portions of the dataset, enabling statistical verification of its generalization success. Thus, it was confirmed that all models neither overfit the training data nor exhibited insufficient learning, but instead generalized reliably in real-world scenarios.

5) *Statistical Significance and Comparative Analysis of the Best Model*

In this study, the statistical significance of the performance differences between the best-performing model—the Hybrid model, whose results are presented in Table XVI—and the next best model, ExtraTrees, was evaluated in Table XVII. In the comparison conducted on the OFF representation, it is observed that all metric differences obtained by the Hybrid model are statistically significant. The standard deviations remaining within the ±0.0004–0.0008 range indicate that both models produce highly stable and reproducible results; the $p < 0.001$ values obtained across the five core metrics further demonstrate that these differences cannot be explained by random chance.

In the statistical significance tests, the differences for the Accuracy, Recall, AUC-ROC, and F1-Score metrics were found to be significant at the $p < 0.001$ level, while the difference in the Precision metric was significant at the $p < 0.01$ level.

Especially the Recall metric (Hybrid: 0.9807 ± 0.0008; ExtraTrees: 0.9681 ± 0.0010; $\Delta = 0.0123$, $p \approx 0.0004$) reveals the most pronounced difference. This finding indicates a meaningful improvement in the model's ability to capture attack samples and reduce false negatives. Considering the critical importance of false negatives in the cybersecurity domain, this superiority in Recall further supports the Hybrid model as the preferable choice for practical use.

In conclusion, the observed performance difference—driven by the Hybrid model's high accuracy (Accuracy = 0.983), strong discriminative power (AUC-ROC = 0.9983), and low variance—is not random; it is statistically significant ($p < 0.001$) and obtained with a high level of reproducibility. These findings demonstrate that, in EMBER-IDS attack analysis, the Hybrid model serves as a reference in terms of reliability and detection consistency.

TABLE XVI: Performance Comparison of Hybrid and ExtraTrees Models (OFF Data Representation, Mean ± Std)

Metric	Hibrit	ExtraTrees
Accuracy	0.9830 ± 0.0004	0.9754 ± 0.0007
Precision	0.9853 ± 0.0006	0.9826 ± 0.0007
Recall	0.9807 ± 0.0008	0.9681 ± 0.0010
AUC-ROC	0.9983 ± 0.0001	0.9972 ± 0.0001
F1-Score	0.9830 ± 0.0004	0.9753 ± 0.0007

TABLE XVII: Difference (Δ) and p-Values Between the Hybrid and ExtraTrees Models

Metric	Δ (Difference)	p-value
Accuracy	0.0076	<0.001
Precision	0.0027	<0.001
Recall	0.0126	$\approx$ 0.0004
AUC-ROC	0.0011	<0.001
F1-Score	0.0077	<0.001

I. Analysis of the Impact of Representations on Model Performance

1) Impact of the PLS-DA Representation on Model Performance

In this study, the impact of the PLS-DA representation—used for dimensionality reduction—on the classification performance of the hybrid model as well as ML and DL models was analyzed.

Performance of the Hybrid Model on the PLS-DA Representation

Table XVIII presents the classification results of the hybrid model on the PLS-DA representation, while Table XIX reports the computational performance indicators such as training time, latency, and throughput. On the PLS-DA representation, the hybrid model achieved a test accuracy of 96.9%, an F1-score of 96.9%, and notably an AUC-ROC of 99.53%, indicating a highly robust performance. These findings demonstrate that the model effectively distinguishes between benign and malicious samples; moreover, the AUC-ROC exceeding 99.5%—a critical metric in cybersecurity applications—confirms that high detection sensitivity is maintained in identifying malware.

The model's computational performance is also highly efficient: with an SAE size of 0.88 MB and an LGBM size of 30.37 MB, the total compact structure of 31.26 MB was trained in only 195.3 seconds, achieving inference with a latency of 0.021 ms/sample and a throughput of 48,140 samples per second.

In contrast, the hybrid model trained on the OFF (raw feature space) representation required a longer training time (493 s), but achieved a significantly faster inference performance with 0.015 ms/sample latency and a throughput of 64,779 samples per second. This outcome can be explained by the model's more efficient optimization when the OFF representation—containing the full raw feature space—is directly provided to the learning process. Additionally, the OFF variant achieved the highest overall accuracy with an AUC of 0.998, and therefore it was considered the primary (reference) result of the study.

In conclusion, although the PLS-DA representation is more compact and computationally faster, the OFF representation demonstrated superior performance in both accuracy and generalization capability, making it the most efficient variant of the hybrid model.

TABLE XVIII: Test Performance Results of the Hybrid Model on PLS-DA Representation

Metric	Result (Test Mean ± Std)
Accuracy	0.9693 ± 0.0006
Precision	0.9738 ± 0.0022
Recall	0.9646 ± 0.0018
Specificity	0.9740 ± 0.0023
F1-Score	0.9692 ± 0.0005
F1-Weighted	0.9693 ± 0.0006
AUC-ROC	0.9953 ± 0.0001

TABLE XIX: Computational Performance Metrics of the Hybrid Model (PLS-DA Representation)

Metric	Value (Mean)	Description
Model Size	31.26 MB ($\approx$ 0.23M parameters)	Total model size
Inference Time	1.663 s ($n = 80\,079$)	Test inference time
Latency	0.021 ms/sample	Average latency
Throughput	48,140 samples/s	Processing speed
Training Time	195.355 s	Average training time

2) Classification Performance of the Hybrid Model on the PLS-DA Representation (Confusion Matrix)

The confusion matrix of the hybrid model on the PLS-DA data representation is presented in Figure 7. Examination of the results shows that the model produced 194,996 correct / 5,202 false positives for the Benign class, and 193,108 correct / 7,091 false negatives for the Malicious class. These values indicate that while the hybrid model provides generally acceptable accuracy on the PLS-DA representation, it exhibits clearly higher error rates compared to the OFF representation.

In particular, the substantially higher number of false negatives (7,091) relative to the OFF representation reveals that PLS-DA does not contribute to the hybrid architecture at the expected level in distinguishing malicious traffic. Similarly, false positive errors also increase compared to the OFF representation. This suggests that because PLS-DA reduces the data structure through a linear projection, it weakens classification capability to some extent, especially in complex and high-variance network traffic.

Nonetheless, the fact that the hybrid architecture is still able to achieve a high number of correct classifications despite the aggressive information loss introduced by a linear projection–based dimensionality reduction method such as PLS-DA demonstrates the strong robustness capacity of the model. Therefore, although the PLS-DA representation is not as successful as OFF, it can be stated that the hybrid architecture still delivers satisfactory and stable performance under this representation.

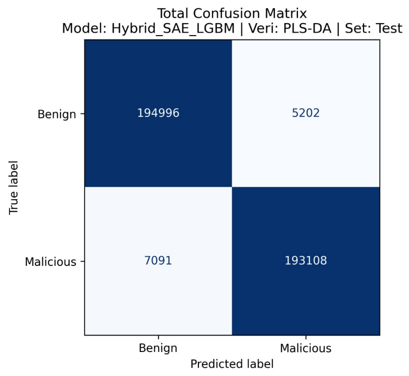

Fig. 7: Confusion Matrix Result of the Hybrid Model on the PLS-DA Representation

Performance of ML Models on the PLS-DA Representation

The comparative performance values of the ML models are presented in Table XX.

When the PLS-DA representation is used, it is observed that the accuracy rates of the machine learning models decrease by approximately 3% to 9% compared to the OFF results. The primary reason for this decrease is that PLS-DA is a linear dimensionality reduction method and cannot fully represent the relationships within the high-dimensional, complex, and non-linear distribution of the EMBER24 dataset. Nevertheless, with appropriate hyperparameter configurations and careful model optimization, high and consistent performance values were also obtained with the PLS-DA representation. In particular, the CatBoost and ExtraTrees models produced strong results in overall accuracy, F1-score, and AUC-ROC metrics even after the PLS-DA transformation; this demonstrates that despite its linear structure, PLS-DA can still provide an effective feature representation on complex datasets when configured properly.

TABLE XX: Test Performance Results of Machine Learning Models on the PLS-DA Representation (Mean ± Std)

Model	Metric	Result (Test Mean ± Std)
CatBoost	Accuracy	0.8936 ± 0.0033
	Precision	0.9045 ± 0.0053
	Recall	0.8801 ± 0.0013
	Specificity	0.9071 ± 0.0056
	F1	0.8922 ± 0.0031
	F1-W	0.8936 ± 0.0033
	AUC-ROC	0.9598 ± 0.0017
ExtraTrees	Accuracy	0.9313 ± 0.0014
	Precision	0.9393 ± 0.0022
	Recall	0.9223 ± 0.0007
	Specificity	0.9403 ± 0.0033
	F1	0.9307 ± 0.0013
	F1-W	0.9313 ± 0.0014
	AUC-ROC	0.9763 ± 0.0005
GradientBoost	Accuracy	0.8795 ± 0.0042
	Precision	0.8901 ± 0.0041
	Recall	0.8658 ± 0.0019
	Specificity	0.8931 ± 0.0041
	F1	0.8788 ± 0.0022
	F1-W	0.8795 ± 0.0042
	AUC-ROC	0.9500 ± 0.0013
HistGB	Accuracy	0.8756 ± 0.0028
	Precision	0.8836 ± 0.0052
	Recall	0.8651 ± 0.0017
	Specificity	0.8860 ± 0.0059
	F1	0.8743 ± 0.0024
	F1-W	0.8756 ± 0.0028
	AUC-ROC	0.9492 ± 0.0016
SGDClassifier	Accuracy	0.5775 ± 0.0281
	Precision	0.6246 ± 0.1048
	Recall	0.5013 ± 0.1687
	Specificity	0.6536 ± 0.2064
	F1	0.5297 ± 0.0835
	F1-W	0.5631 ± 0.0275
	AUC-ROC	0.5775 ± 0.0281

TABLE XXI: Test Performance Results of Deep Learning Models on the PLS-DA Representation (Mean ± Std)

Model	Metric	Result (Test Mean ± Std)
DNN	Accuracy	0.9630 ± 0.0013
	Precision	0.9702 ± 0.0018
	Recall	0.9553 ± 0.0031
	Specificity	0.9707 ± 0.0021
	F1	0.9627 ± 0.0013
	F1-W	0.9630 ± 0.0013
	AUC-ROC	0.9935 ± 0.0003
MLP-Mixer	Accuracy	0.9625 ± 0.0025
	Precision	0.9694 ± 0.0025
	Recall	0.9552 ± 0.0036
	Specificity	0.9699 ± 0.0026
	F1	0.9618 ± 0.0006
	F1-W	0.9625 ± 0.0025
	AUC-ROC	0.9931 ± 0.0003
ResNetMLP	Accuracy	0.9629 ± 0.0019
	Precision	0.9707 ± 0.0020
	Recall	0.9547 ± 0.0023
	Specificity	0.9712 ± 0.0019
	F1	0.9626 ± 0.0011
	F1-W	0.9629 ± 0.0019
	AUC-ROC	0.9934 ± 0.0004
Wide&Deep	Accuracy	0.9623 ± 0.0015
	Precision	0.9686 ± 0.0026
	Recall	0.9585 ± 0.0020
	Specificity	0.9661 ± 0.0027
	F1	0.9621 ± 0.0015
	F1-W	0.9623 ± 0.0015
	AUC-ROC	0.9915 ± 0.0005
gMLP	Accuracy	0.9636 ± 0.0016
	Precision	0.9699 ± 0.0028
	Recall	0.9569 ± 0.0028
	Specificity	0.9703 ± 0.0028
	F1	0.9628 ± 0.0014
	F1-W	0.9636 ± 0.0016
	AUC-ROC	0.9936 ± 0.0004

Performance of DL Models on the PLS-DA Representation

When examining the performance results of the deep learning models under the PLS-DA representation presented in Table XXI, it is observed that the accuracy values decrease by only about 1–1.5% compared to the OFF case. The main reason this difference remains limited is that the deep learning models were reconfigured with optimal hyperparameters, and their learning capacity was increased in a way that compensates for the information loss introduced by the representation transformation. Under normal circumstances, a linear representation method such as PLS-DA would be expected to cause a more pronounced decrease in performance, since it cannot fully capture the complex and non-linear relationships in the data. However, in this study, high generalization capability and consistent accuracy values were obtained even with the PLS-DA representation, particularly due to the careful optimization of models such as MLP-Mixer and gMLP.

On the other hand, although the hybrid architecture achieved slightly lower accuracy—approximately a 0.5-point difference—than the deep learning models under the PLS-DA representation, it operated significantly more efficiently in terms of computational cost and achieved similar accuracy levels in a shorter time and with lower resource usage. This demonstrates the overall efficiency of the hybrid architecture and its superiority in terms of cost–performance balance at the application scale.

Overall, considering that PLS-DA is a linear dimensionality reduction method, it remains limited in its ability to represent non-linear relationships, particularly in attack patterns. For this reason, a decrease of approximately 2–6% in accuracy was observed in ML models. In contrast, DL models were able to largely compensate for this information loss due to their high generalization capacity, resulting in only about a 1% reduction in accuracy.

The proposed Hybrid model demonstrated the highest overall performance even under the PLS-DA representation, achieving 96.9% accuracy and 99.5% AUC-ROC. This performance highlights not only high accuracy but also very low variance (Accuracy std ±0.0006, AUC-ROC std ±0.0001), indicating that the model is extremely stable.

The decrease in accuracy and AUC-ROC values under the PLS-DA representation remained limited to only 1–3% compared to the raw variant; this demonstrates that DL models, in particular, can effectively compensate for the information loss caused by linear representation. Although the decrease in ML models was more pronounced, CatBoost and ExtraTrees still achieved statistically significant high performance ($p < 0.01$).

In conclusion, the Hybrid model produced the most stable results with the lowest variance even under the PLS-DA representation; the achieved performance is not random but statistically strong ($p < 0.001$) and highly reproducible.

*3) **Impact of the VAE Representation on Model Performance***

Performance of the Hybrid Model on the VAE Representation

According to the performance of the hybrid model on the VAE representation presented in Table XXII, the model is observed to have maintained its highest overall performance. Compared to the raw data case, the decrease in accuracy is only about 0.0016, which is statistically negligible. This result indicates that the hybrid model preserves its generalization capability without experiencing information loss, as it can effectively represent complex and non-linear data relationships under the VAE representation. Consequently, the hybrid architecture (SAE + LGBM) demonstrated consistent and stable performance on both the VAE and OFF representations, achieving high accuracy (≈ 0.98), strong AUC-ROC (≈ 0.998), and low variance values. This finding methodologically confirms that the hybrid approach offers high generalization capacity and computational efficiency, independent of the data representation.

TABLE XXII: 5-Fold Cross-Validation Results of the Hybrid Model on the VAE Representation

Metric	Result (Test Mean ± Std)
Accuracy	0.9814 ± 0.0005
Precision	0.9853 ± 0.0012
Recall	0.9775 ± 0.0016
Specificity	0.9854 ± 0.0013
F1-Score	0.9814 ± 0.0005
F1-Weight	0.9814 ± 0.0005
AUC-ROC	0.9981 ± 0.0001

4) *Classification Performance of the Hybrid Model on the VAE Representation (Confusion Matrix)*

The confusion matrix of the hybrid model for the VAE data representation is presented in Figure 8. Upon examining the matrix, it is observed that the model produced 197,279 correct / 2,919 false positives for the Benign class, and 195,687 correct / 4,512 false negatives for the Malicious class. Although this distribution is generally balanced, comparison with the results obtained under the OFF representation reveals that the hybrid model produces higher error rates on the VAE representation, indicating a noticeable decline in attack detection performance.

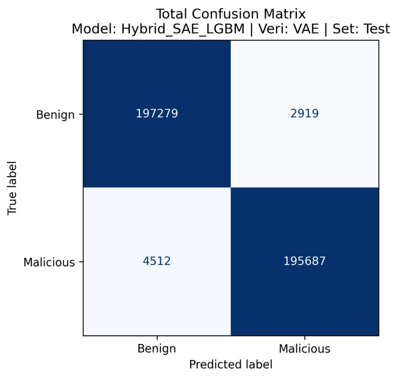

Fig. 8: Confusion Matrix Result of the Hybrid Model on the VAE Representation

Compared to the OFF representation, the hybrid model maintained extremely low false positive and false negative values, demonstrating its strongest performance. Under the VAE representation, however, the increase in error counts indicates a weaker performance—particularly in distinguishing attack traffic—relative to the OFF representation. Nevertheless, considering that VAE provides a much more compressed and lower-dimensional representation, it is noteworthy that the hybrid architecture still produced high accuracy and balanced classification performance under this representation.

In conclusion, although the VAE representation presents a more challenging scenario than OFF, the hybrid model can still be considered successful, stable, and practically applicable even under this representation.

Performance of ML Models on the VAE Representation

Table XXIII presents the results of the machine learning models on the VAE representation. Compared to the OFF case, noticeable but acceptable decreases in accuracy were observed. The primary reason for this difference is that VAE represents features in a probabilistic manner, which partially smooths some of the discriminative information. Nevertheless, the CatBoost and ExtraTrees models maintained strong generalization performance, achieving accuracy ($\approx$0.88–0.93) and AUC-ROC ($\approx$0.95–0.97) values. This indicates that, when properly configured, VAE representations still provide sufficient information density for tree-based algorithms.

On the other hand, the SGDClassifier model produced low results in both the OFF and VAE representations; its accuracy ($\approx$0.52) and F1 ($\approx$0.22) values clearly demonstrate that this model is not suitable for the given dataset and representation structure. In conclusion, under the VAE representation, machine learning models exhibited generally stable but more limited performance compared to raw data; however, satisfactory results were obtained in terms of computational efficiency and the balance between generalization and performance.

TABLE XXIII: Test Performance Results of Machine Learning Models on the VAE Representation (Mean ± Std)

Model	Metric	Result (Test Mean ± Std)
CatBoost	Accuracy	0.8810 ± 0.0014
	Precision	0.8969 ± 0.0021
	Recall	0.8610 ± 0.0018
	Specificity	0.9010 ± 0.0023
	F1	0.8786 ± 0.0014
	F1-W	0.8810 ± 0.0014
	AUC-ROC	0.9525 ± 0.0008
ExtraTrees	Accuracy	0.9265 ± 0.0015
	Precision	0.9418 ± 0.0027
	Recall	0.9093 ± 0.0015
	Specificity	0.9438 ± 0.0028
	F1	0.9253 ± 0.0014
	F1-W	0.9265 ± 0.0015
	AUC-ROC	0.9771 ± 0.0006
GradientBoost	Accuracy	0.8614 ± 0.0019
	Precision	0.8693 ± 0.0030
	Recall	0.8507 ± 0.0042
	Specificity	0.8721 ± 0.0037
	F1	0.8599 ± 0.0020
	F1-W	0.8614 ± 0.0019
	AUC-ROC	0.9387 ± 0.0015
HistGB	Accuracy	0.8604 ± 0.0015
	Precision	0.8692 ± 0.0038
	Recall	0.8486 ± 0.0023
	Specificity	0.8723 ± 0.0045
	F1	0.8588 ± 0.0011
	F1-W	0.8604 ± 0.0015
	AUC-ROC	0.9391 ± 0.0008
SGDClassifier	Accuracy	0.5209 ± 0.0186
	Precision	0.6043 ± 0.1016
	Recall	0.1382 ± 0.0249
	Specificity	0.9036 ± 0.0418
	F1	0.2231 ± 0.0328
	F1-W	0.4382 ± 0.0187
	AUC-ROC	0.6071 ± 0.0758

Performance of DL Models on the VAE Representation

Table XXIV presents the performance results of the deep learning models on the VAE representation. Compared to the OFF (raw data) case, a decrease of approximately 2–2.5% in accuracy was observed; however, this difference was not at a level that would weaken generalization performance. The primary reason for this decrease is that VAE transforms the data into a probabilistic latent space, which partially softens the deterministic relationships among features.

Nevertheless, models such as Wide&Deep, MLP-Mixer, and gMLP still exhibited high performance, achieving accuracy (≈0.94–0.95) and AUC-ROC (≈0.988–0.989) values that surpass many studies reported in the literature. This result indicates that, thanks to the VAE's ability to capture non-linear data distributions, the models maintain strong generalization capability despite information loss.

Overall, deep learning models experienced a slight decrease in accuracy under the VAE representation, yet maintained stable, consistent, and high AUC-ROC performance. These findings methodologically confirm that when optimized with appropriate hyperparameters, VAE serves as a supportive mechanism that enhances the representation learning capacity of deep models.

TABLE XXIV: Test Performance Results of Deep Learning Models on the VAE Representation (Mean ± Std)

Model	Metric	Result (Test Mean ± Std)
DNN	Accuracy	0.9462 ± 0.0099
	Precision	0.9566 ± 0.0106
	Recall	0.9349 ± 0.0114
	Specificity	0.9575 ± 0.0105
	F1	0.9456 ± 0.0101
	F1-W	0.9462 ± 0.0099
	AUC-ROC	0.9886 ± 0.0035
MLP-Mixer	Accuracy	0.9486 ± 0.0080
	Precision	0.9597 ± 0.0021
	Recall	0.9365 ± 0.0161
	Specificity	0.9607 ± 0.0021
	F1	0.9479 ± 0.0086
	F1-W	0.9485 ± 0.0080
	AUC-ROC	0.9890 ± 0.0029
ResNetMLP	Accuracy	0.9469 ± 0.0084
	Precision	0.9587 ± 0.0062
	Recall	0.9341 ± 0.0141
	Specificity	0.9597 ± 0.0061
	F1	0.9462 ± 0.0088
	F1-W	0.9469 ± 0.0084
	AUC-ROC	0.9889 ± 0.0033
Wide&Deep	Accuracy	0.9495 ± 0.0080
	Precision	0.9586 ± 0.0037
	Recall	0.9395 ± 0.0133
	Specificity	0.9595 ± 0.0033
	F1	0.9490 ± 0.0084
	F1-W	0.9495 ± 0.0080
	AUC-ROC	0.9884 ± 0.0028
gMLP	Accuracy	0.9493 ± 0.0077
	Precision	0.9590 ± 0.0055
	Recall	0.9387 ± 0.0118
	Specificity	0.9599 ± 0.0053
	F1	0.9487 ± 0.0080
	F1-W	0.9493 ± 0.0077
	AUC-ROC	0.9892 ± 0.0028

Time-Based Model Performance Analysis

(Temporal Evaluation Experiments)

Time-Based Experimental Design, Model Selection, and Temporal Drift Analysis

The primary objective of the temporal experiment is to predict future instances by leveraging past data. In this approach, the model learns exclusively from historical observations, thereby providing a realistic time-series generalization. Since random splitting methods disrupt the temporal order of the data, they may introduce a serious risk of data leakage in temporal scenarios; in contrast, temporal splitting preserves the chronological structure and completely eliminates this risk. Maintaining the temporal order at the file or week level is critically important in domains such as malware detection, where threats evolve continuously. This ensures that the model's generalization capability against future, previously unseen attack samples can be evaluated in a reliable manner.

In this study, a temporal drift analysis was conducted to assess the model's robustness against time-dependent changes in the data distribution. During the 5-fold cross-validation phase, all machine learning (ML) and deep learning (DL) models were tested on three different data representations (OFF, PLS, and VAE), and a comprehensive performance comparison was performed. In the temporal analysis phase, instead of retraining all models, only the two best-performing ML models (ExtraTrees and CatBoost), two DL models (DNN and gMLP), and the proposed Hybrid model were selected. This selection was made to enhance computational efficiency and to evaluate the performance of practically applicable, high-generalization models under time-based scenarios.

In the temporal analysis, instead of random cross-validation (K-Fold) methods that may disrupt temporal order, a single hold-out splitting strategy that preserves time dependency was applied. The dataset was sorted according to weekly chronological order, and this structure was verified through the JSONL files.

The training set covers the first 53 weeks. This period was validated using the following JSONL records:

- First week (Train): `2023-09-24_2023-09-30_Win32_train.jsonl`
- Last week (Train): `2024-09-15_2024-09-21_Win32_train.jsonl`

This interval contains a total of 200,000 samples (Benign: 100,000, Attack: 100,000).

The temporal test set covers the 12 weeks immediately following the training period. This period was validated using the following JSONL records:

- First week (Test): `2024-09-22_2024-09-28_Win32_test.jsonl`
- Last week (Test): `2024-12-08_2024-12-14_Win32_test.jsonl`

The temporal test set contains a total of 199,990 samples (Benign: 99,994, Attack: 99,996).

This design ensures that no temporal leakage occurs between the training and test sets, resulting in a realistic future-prediction scenario. The class distribution of the training and temporal test sets is presented in Figure 9.

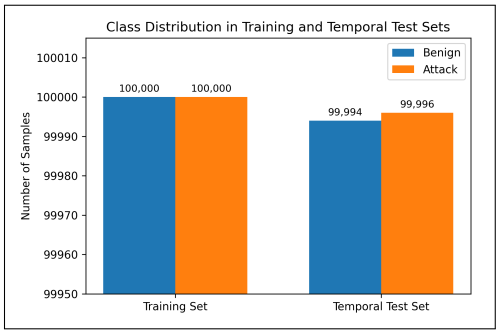

Fig. 9: Class distribution in the training and temporal test sets used in the temporal experiment

All data preprocessing and feature extraction steps used in the 5-fold cross-validation experiments were applied identically to the temporal subsets. The training and test sets were cleaned independently, and variance and correlation analyses were performed separately for each subset. In the temporal test set, 37 low-variance columns and 110 highly correlated feature pairs were identified; as a result of this analysis, 42 features were removed from the dataset due to high correlation.

In the temporal scenario, the processing pipeline of the hybrid model is presented in Figure 10. This workflow comprises the transformation of features—obtained after StandardScaler-based normalization and variance thresholding—into a latent representation using a Stacked Autoencoder, while simultaneously enriching the feature space through gain-based Top-K feature selection. The final representation, formed by concatenating both latent and selected features, is then passed to the LightGBM classifier, where the temporal test set is evaluated. This architecture aims to enhance the representational robustness of the model against feature drift occurring in temporal data streams and thereby improve time-dependent generalization performance.

In the temporal scenario, all models were evaluated using the default threshold of 0.5, whereas an additional decision threshold optimization was performed exclusively for the hybrid model. The threshold was first selected using a multi-metric optimization objective; if this objective could not be satisfied, the Youden's J statistic (TPR–FPR) was employed to identify the threshold yielding the highest overall performance. Thus, decision boundary optimization was intentionally designed as a hybrid model–specific enhancement.

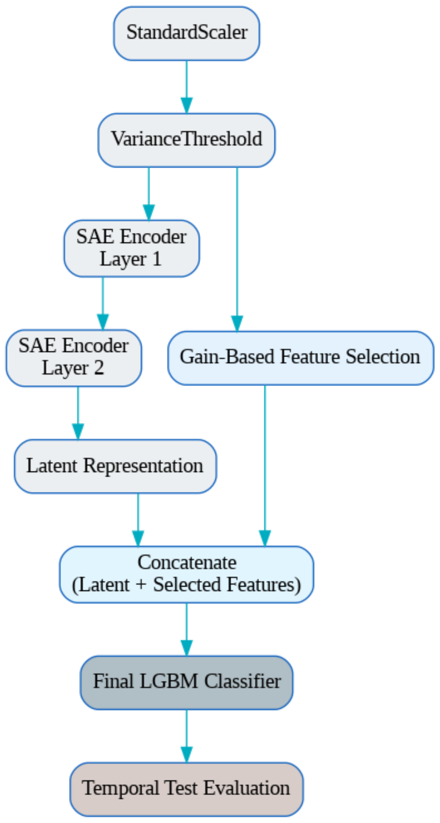

Fig. 10: Processing pipeline of the hybrid SAE–Top-K–LGBM model in the temporal scenario

Feature Alignment of Training and Test Attributes in Temporal Drift Analysis

Since the training and test sets originate from different weeks in the temporal scenario, a natural temporal shift (feature drift) has emerged in the feature space. Specifically, while the training set contains 2,504 features, the temporal test set retains only 2,493 features after preprocessing; this difference indicates that some features either lost their discriminative power over time or began to exhibit higher correlation, leading to their removal from the dataset.

For this reason, during the coding phase, both the training and test datasets were read from parquet files, all features were converted to the `float32` type, and column names were compared. Features that existed in the training set but were missing in the test set were added to the test set with a neutral initialization value of 0.0, while columns unique to the test set were removed. Subsequently, the column ordering of the test set was made identical to that of the training set, and the column count and data types were verified through consistency checks before the aligned test set was finalized.

This process ensured that the feature space remained consistent in the temporal experiment without violating the separation between training and test sets. Furthermore, it highlights the potential need for model retraining or adaptation mechanisms when dealing with future data exhibiting temporal drift.

Hyperparameter Adaptation in the Temporal Drift Scenario (ML, DL, and Hybrid Models)

Since the temporal scenario differs distributionally from the 5-fold experimental setup, the models were re-optimized specifically for this stage. Using the same hyperparameters on time-dependent datasets is not methodologically appropriate, as the optimal settings vary depending on the severity and structure of distributional shifts (drift). Therefore, the core hyperparameters used in the temporal experiments were redefined for each model family, and a summarized overview is provided in Table XXV. The table systematically presents the temporal-condition–adapted core configurations of the ML, DL, and hybrid models.

TABLE XXV: Core hyperparameters of the Hybrid, ML, and DL models used in the temporal test scenario.

Model	**Parameter**	**Value**
Hybrid Model (SAE + Top-K + LGBM)		
Hybrid	SAE latent_dim	192
Hybrid	Top-K features	256
Hybrid	LGBM n_estimators	6000
Machine Learning Models		
CatBoost	iterations	150
CatBoost	learning_rate	0.3
ExtraTrees	n_estimators	200
ExtraTrees	max_depth	12
Deep Learning Models		
DNN	hidden_units	(512, 256, 128, 64)
DNN	dropout	0.3
gMLP-Tabular	blocks	2
gMLP-Tabular	dim_ff	256

Comparative Performance Analysis of Models in the Temporal Drift Scenario

When Table XXVI and Figure 11, which present the results of the top three performing models, are examined, it is clearly observed that the hybrid model demonstrates a more balanced and higher performance compared to classical machine learning–based models (CatBoost and ExtraTrees). In particular, the hybrid architecture exhibits strong generalization capability both in overall accuracy and in sensitivity to the positive (attack) class, as evidenced by its distinctly higher Accuracy (96.70%), Recall (96.10%), F1 (96.68%), F1-Weighted (96.70%), and AUC-ROC (99.53) values compared to the other models. Furthermore, Table XXVII, which provides additional error rates and complementary validation metrics for the hybrid model, supports this assessment by demonstrating that the model maintains a stable error profile with low FPR and FNR values.

TABLE XXVI: Performance Results of All Models in the Temporal Test Scenario

Model	Acc	Rec	Spec	Prec	F1	F1W	AUC
Hybrid	0.9670	0.9610	0.9730	0.9726	0.9668	0.9670	0.9953
CatBoost	0.9574	0.9450	0.9698	0.9691	0.9569	0.9574	0.9933
ExtraTrees	0.9391	0.9255	0.9528	0.9514	0.9383	0.9391	0.9889
DNN	0.9210	0.9677	0.8744	0.8851	0.9246	0.9209	0.9818
gMLP	0.9143	0.8859	0.9426	0.9391	0.9118	0.9142	0.9762

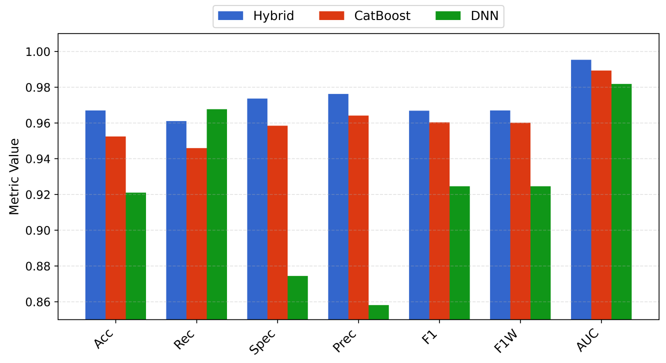

Fig. 11: Bar Chart Comparison of the Hybrid, CatBoost, and DNN Models in the Temporal Test Scenario.

TABLE XXVII: Additional Error Rates and Accuracy Metrics of the Hybrid Model (Temporal Test)

Metric	Description	Value (%)
FPR	False Positive Rate	2.698
FNR	False Negative Rate	3.870
NPV	Negative Predictive Value	96.14
FDR	False Discovery Rate	2.71

CatBoost and ExtraTrees models exhibited a limited advantage in terms of Precision and Specificity; however, these differences are statistically negligible (less than 1%). More importantly, the superiority of the hybrid model in critical attack-detection metrics such as Recall and AUC-ROC is far more meaningful in real-world scenarios, particularly regarding the reduction of false negatives (i.e., undetected attacks). Therefore, in the context of the model's primary objective—capturing attacks as completely as possible—the hybrid architecture is more advantageous.

Although classical tree-based models show minor advantages in certain sub-metrics (e.g., precision, specificity), the hybrid architecture produced more consistent and superior results in terms of overall performance, imbalance tolerance, and long-term stability. This supports the fundamental argument that hybrid architectures enhanced with deep feature extraction offer greater generalization capability against evolving attack patterns compared to classical methods.

Within the scope of temporal analysis, deep learning models were also evaluated, and both DNN and gMLP exhibited significantly lower performance than the hybrid model. The DNN model produced 92.10% accuracy and 92.46% F1-score in the temporal test, while showing a marked decline in Specificity (87.44%), which represents its ability to distinguish benign samples. Similarly, the gMLP model yielded weaker results with 91.43% accuracy and 88.59% recall. These findings indicate that DL models are more sensitive to temporal shifts and changes in data distribution.

In the hybrid model, the Stacked Autoencoder-based feature extraction and the subsequent LGBM classifier better capture the complex distributions of the data and provide flexible adaptation to time-dependent distribution shifts (concept drift). Consequently, the hybrid architecture demonstrates not only high instantaneous accuracy but also superior long-term temporal robustness.

Overall, the findings indicate that the hybrid approach has strong potential for delivering sustainable performance in attack detection, scientifically justifying its positioning as a more advanced solution compared to traditional methods.

Comparison of Confusion Matrices Across Models on the Temporal Test Set

When examining the confusion matrix results for the temporal test set (Figure 12 – Figure 16), it is evident that the hybrid SAE+LGBM architecture exhibits a much more balanced and consistent classification performance compared to all other models. The hybrid model produced 97,296 correct classifications and only 2,698 false positives for the Benign class; for the Malicious class, it achieved 96,100 correct detections with 3,896 false negatives. This distribution indicates that errors are both low and balanced across benign and attack classes, demonstrating that the model achieves a strong equilibrium in terms of both precision and recall. In particular, the low number of false negatives constitutes a critical advantage for attack detection.

Examining the confusion matrix of the CatBoost model shows 96,982 correct / 3,012 incorrect for the Benign class and 94,499 correct / 5,497 incorrect for the Malicious class. Although the model delivers generally high accuracy, both false positives and false negatives are higher than those of the hybrid model. The increased error rate in the attack class (TP: 94,499 — approximately 1,600 fewer than the hybrid) indicates that CatBoost has more limited generalization capacity under temporal drift.

The performance decline of the ExtraTrees model is even more pronounced: 95,275 correct / 4,719 incorrect for the Benign class and 92,549 correct / 7,447 incorrect for the Malicious class. These values indicate a substantially higher number of false negatives in the attack class, revealing that ExtraTrees responds weakly to distributional shifts (drift) in the temporal scenario and suffers significant performance degradation in detecting malicious traffic.

For the deep learning–based gMLP model, the drift effect is even more apparent. The model produced 94,252 correct / 5,742 incorrect for Benign and 88,594 correct / 11,402 incorrect for Malicious, showing a substantial increase in both false positive and false negative rates. The 11,402 false negatives, in particular, demonstrate that gMLP struggles severely under temporal conditions. This value corresponds to nearly three times more missed attacks compared to the hybrid model.

The confusion matrix results of the DNN model are even less favorable than those of gMLP. With 87,440 correct / 12,554 incorrect for Benign and 96,767 correct / 3,229 incorrect for Malicious, the model exhibits a dramatic increase in false positives within the benign class. DNN has difficulty distinguishing benign traffic in the temporal test and produces a high number of false positives. This confirms that the model is highly sensitive to temporal data drift.

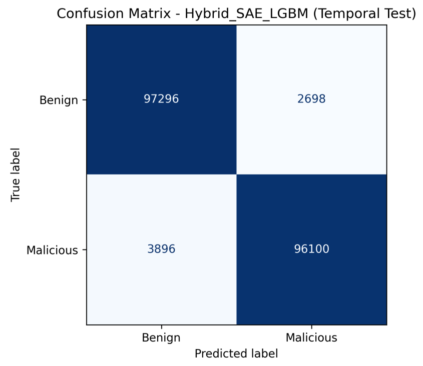

Fig. 12: Confusion Matrix Result of the Hybrid Model on the Temporal Test Set

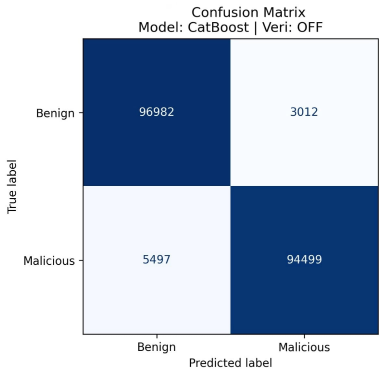

Fig. 13: Confusion Matrix Result of the CatBoost Model on the Temporal Test Set

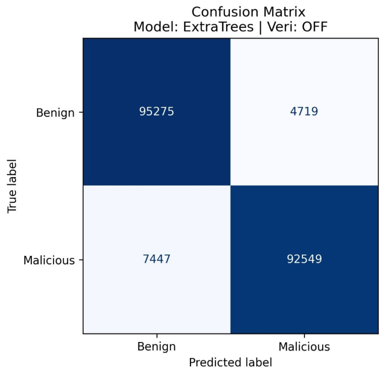

Fig. 14: Confusion Matrix Result of the ExtraTrees Model on the Temporal Test Set

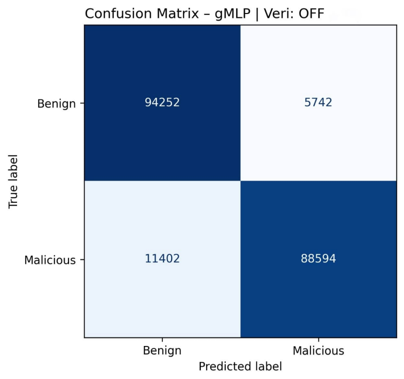

Fig. 15: Confusion Matrix Result of the gMLP Model on the Temporal Test Set

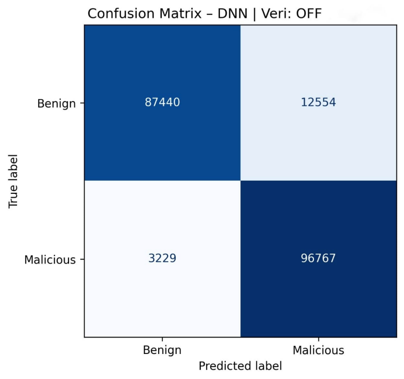

Fig. 16: Confusion Matrix Result of the DNN Model on the Temporal Test Set

Model Size and Computational Performance Metrics of the Hybrid Architecture in Temporal Analysis

In the temporal test scenario, the computational performance of the hybrid model indicates that the proposed architecture is remarkably efficient, featuring a highly compact model size of only 21.30 MB and approximately 3.03 million trainable parameters. The model successfully processed 199,990 instances within a total inference time of 1.664 seconds, yielding an extremely low latency of 0.008 ms per sample. Furthermore, the throughput value of 120,180 samples/s demonstrates that the architecture is capable of handling well over one hundred thousand flow records per second, confirming that the hybrid model fully satisfies the high data processing requirements of real-time IDS systems. These computational metrics are summarized in Table XXVIII.

When these indicators are evaluated collectively, it becomes evident that the hybrid architecture not only provides high accuracy but also offers a lightweight, highly optimization-friendly, and computationally efficient model structure. Thanks to its low latency, high throughput, and compact model size, the architecture delivers maximum performance with minimal resource consumption in practical scenarios where computational cost is critical—such as real-time network traffic monitoring, edge-device integrations, and enterprise IDS deployments. The simultaneous achievement of low latency, high throughput, and a compact model size clearly demonstrates that the proposed approach constitutes a robust and sustainable solution not only in terms of accuracy but also in terms of computational cost, speed, and real-time usability.

Additionally, the hybrid architecture's low-complexity, optimization-ready, and hardware-efficient design—its ability to deliver high performance even without GPU support, and its low parameter footprint enabling stable operation on resource-constrained devices—further enhances its operational efficiency in real-world environments. With these characteristics, the hybrid model offers a scalable, lightweight, and high-performance threat detection solution suitable for both academic research and practical deployments.

TABLE XXVIII: Computational Performance Metrics of the Hybrid Model on the Temporal Test Set

Metric	Value (Average)	Description
Model Size	21.30 MB ($\approx$ 3.03M parameters)	Total model size
Inference Time	1.664 s (n = 199,990)	Test inference time
Latency	0.008 ms/sample	Average latency
Throughput	120,180 samples/s	Processing speed
Training Time	71.151 s	Total training time

Class-Based ROC Curve of the Hybrid Model in Temporal Analysis

The ROC curve presented in Figure 17 clearly demonstrates the discriminative power of the hybrid SAE+LGBM model on the temporal test set. The curve exhibits near-ideal behavior (close to the upper-left corner) by achieving high true positive rates even at low false positive rates. The obtained AUC-ROC value of 0.9954 confirms that the hybrid model delivers an almost perfect performance in distinguishing the Malicious class from benign traffic. This high AUC value indicates that the model is not only effective at a specific threshold but also maintains consistent and reliable discriminative capability across all possible decision thresholds. The ROC curve's strong alignment with the ideal shape, even under temporal drift, further demonstrates that the hybrid architecture provides robust generalization against time-evolving attack distributions.

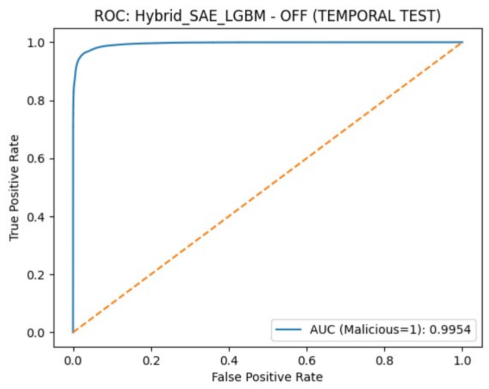

Fig. 17: ROC Curve of the Hybrid Model in Temporal Analysis

References

[1] Svante Wold, Michael Sjöström, and Lennart Eriksson. Pls-regression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems*, 58(2):109–130, 2001.

[2] Stephen Odaibo. Tutorial: Deriving the standard variational autoencoder (vae) loss function. *arXiv preprint arXiv:1907.08956*, 2019.

[3] Tim Silhan, Stefan Oehmcke, and Oliver Kramer. Evolution of stacked autoencoders. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 823–830. IEEE, 2019.

[4] David Durfee and Ryan M Rogers. Practical differentially private top-k selection with pay-what-you-get composition. *Advances in Neural Information Processing Systems*, 32, 2019.

[5] Hualong Liao, Xinyuan Zhang, Can Zhao, Yu Chen, Xiaoxi Zeng, and Huafeng Li. Lightgbm: an efficient and accurate method for predicting pregnancy diseases. *Journal of Obstetrics and Gynaecology*, 42(4):620–629, 2022.

[6] Payal Awwal and Smita Naval. Development of heuristic adapted serial-based deep learning for efficient adversarial malware detection framework in windows. 326:114032.