Received: 23 August 2024 | Revised: 3 June 2025 | Accepted: 15 July 2025 | Published online: 26 August 2025

RESEARCH ARTICLE

Artificial Intelligence and Applications
2025, Vol. 3(4) 428-442
DOI: 10.47852/bonviewAIA52024151

BON VIEW PUBLISHING

Using Variational Autoencoders with Machine Learning Algorithms in Cyber Security Applications

Thomas Taylor[1], Amna Eleyan[1] and Mohammed Al-Khalidi[1,*]

[1] *Department of Computing and Mathematics, Manchester Metropolitan University, UK*

Abstract: In the evolving field of cybersecurity, detecting malicious activity in high-dimensional network data remains a persistent challenge for traditional machine learning (ML) techniques. This study investigates the use of convolutional variational autoencoders (VAEs) to generate latent features that enhance the performance of various ML classifiers on the 2015 NSL-KDD dataset. Classifiers, including Gaussian Naïve Bayes (GNB), support vector machines (SVMs) with Radial Basis Function (RBF) kernel, decision trees, and dense neural networks, were evaluated using metrics such as accuracy, precision, recall, F1 score, and the Matthews Correlation Coefficient (MCC). To assess the effectiveness of VAEs, Principal Component Analysis (PCA) was used as a baseline dimensionality reduction method, and performance comparisons were made. The best-performing model was an SVM with an RBF kernel, a PCA (threshold = 0.92), and a VAE with six latent features, achieving an accuracy of 82.8%, an F1 score of 0.830, and an MCC of 0.682. The results indicate that VAEs can significantly enhance classifier performance, particularly in GNB and SVM models, suggesting their value in developing more effective intrusion detection systems.

Keywords: machine learning, cybersecurity, autoencoder, neural network, SVM, variational autoencoders, malware

1. Introduction

There is growing interest in the application of machine learning (ML) to cybersecurity, where it is increasingly used to detect patterns and anomalies in large-scale network data [1–3]. ML techniques have already shown effectiveness in diverse domains such as phishing detection, agriculture [4], and healthcare [5], thanks to their ability to classify complex behaviors and improve predictive accuracy. However, in the realm of intrusion detection systems (IDSs), many existing approaches still rely on shallow learning models or traditional dimensionality reduction (DR) methods, such as Principal Component Analysis (PCA). These conventional techniques often struggle to generalize to new or evolving attack types, especially when operating on high-dimensional, noisy network traffic data. Neural networks (NNs), particularly autoencoders, offer a more powerful framework for representation learning by reducing dimensionality while retaining critical features. Among them, convolutional variational autoencoders (VAEs) stand out for their ability to capture both spatial and statistical dependencies in data, offering a probabilistic approach to feature learning. Despite their potential, VAEs remain underexplored in network intrusion detection, where most prior work has focused on standard autoencoders or PCA. The generative nature of VAEs enables them to learn robust, generalizable representations that may be more effective at identifying unseen attack patterns. This study addresses this gap by investigating the use of convolutional VAEs to generate latent feature representations and evaluating their impact on the performance of various ML classifiers. Using the NSL-KDD dataset, a widely used benchmark for IDS research, we compare VAE-generated features with those produced by PCA, as well as with hybrid PCA-VAE transformations. We evaluate classifier performance using multiple metrics and across several algorithms, including Gaussian Naïve Bayes (GNB), support vector machines (SVMs), decision trees, and NNs. This research contributes to the growing body of work on intelligent intrusion detection by offering a comprehensive analysis of DR techniques in IDS pipelines. It not only assesses the predictive gains from using VAEs but also provides practical insights into which DR-classifier combinations yield the best results. By doing so, the study offers both theoretical and practical advancements that can inform the design of more effective and adaptable IDS solutions. The remainder of this paper is structured as follows: Section 2 reviews related work in the domain of ML-based intrusion detection and DR techniques. Section 3 presents the theoretical background, covering the NSL-KDD dataset, PCA, and VAEs. Section 4 outlines the experimental methodology, including data preprocessing steps, DR configurations, and classifier training procedures. Section 5 presents the results and discussion, analyzing the performance of different DR strategies across multiple classifiers, and highlights key findings and limitations. Finally, Section 6 concludes the study and suggests directions for future research.

2. Related Work

Although ML research has been extensive, limited attention has been given to the specific application of autoencoders in improving classification algorithms within cybersecurity contexts. One influential paper that inspired this work is by He et al. [6], which employs a standard autoencoder to create latent features and assesses the accuracy of different ML algorithms both with and without these features using the NSL-KDD dataset. The study shows improvements of 1.00% to 6.74% in classification accuracy for algorithms such as GNB, SVMs, and XGBoost. However, it does not explore advanced feature extraction

***Corresponding author:** Mohammed Al-Khalidi, Department of Computing and Mathematics, Manchester Metropolitan University, UK. Email: M.Al-Khalidi@mmu.ac.uk

techniques like VAEs or compare multiple DR approaches. Al-Qatf et al. [7] train SVMs on latent features generated through an autoencoder, also using the NSL-KDD dataset. The method aims to reduce SVM training time and improve adaptability to new threats. Although the model achieves strong results, 84.96% for binary classification and 80.48% for multiclass, the autoencoder used is a standard one, lacking a probabilistic formulation that may be beneficial for generalization. In Ahmad et al. [8], a recurrent NN (RNN) is trained on NSL-KDD, leveraging temporal dependencies to detect patterns such as DDoS attack signatures. The RNN reaches 83.28% binary and 81.29% multiclass accuracy but requires significant computational resources and careful tuning, which may limit scalability in real-world applications. Xu et al. [9] employ a five-layer autoencoder combined with various preprocessing techniques for anomaly detection. The approach measures reconstruction error between input and output to detect attacks, achieving 90.61% accuracy and a 92.26% F1 score. However, it focuses on reconstruction error rather than classification, limiting its direct applicability for supervised learning tasks. In the work of Song et al. [10], the authors explore varying autoencoder network sizes across different datasets, including NSL-KDD, and apply the resulting models in practical IoT environments. Although demonstrating scalability and adaptability, the paper does not provide a controlled comparison of autoencoder features versus PCA or VAEs across different classifiers. Although these studies demonstrate promising performance using autoencoders and traditional ML classifiers, they also exhibit certain limitations. Standard autoencoders often lack probabilistic modeling capabilities, which may reduce their generalization to unseen attack types. PCA, though computationally efficient, is limited to linear transformations and does not capture complex relationships in data. Moreover, comparisons across studies are often inconsistent due to differences in evaluation metrics, model tuning, and dataset handling. To highlight the differences among methods applied to the same dataset, a comparative summary is provided in Table 1.

In addition to the studies previously mentioned, several recent works have proposed advanced ML frameworks for intrusion detection. In Rabie et al. [11], a novel IoT IDS using Decisive Red Fox optimization with a descriptive back-propagated Radial Basis Function (RBF) network has shown strong adaptability to real-time environments. Similarly, research by Prashanth et al. [12] on optimal feature selection using evolutionary algorithms has demonstrated significant performance gains by reducing input redundancy. Another innovative approach utilizes a Perceptual Pigeon Galvanized Optimization (PPGO) framework in combination with a Likelihood Naïve Bayes classifier for improved attack classification accuracy [13]. Furthermore, optimization-based detection using WI-CS and GNN algorithms in SCADA networks presents valuable advances in critical infrastructure security [14]. Lastly, broad surveys such as Mummadi et al. [15] offer comprehensive insights into ML applications in cybersecurity. These studies highlight a growing trend toward optimization-enhanced and biologically inspired models in IDS, distinguishing our work by focusing on the comparative analysis of probabilistic latent-space learning (via VAEs) against conventional DR in improving classifier performance.

Building on these findings, this paper explores the effectiveness of convolutional VAEs, a probabilistic and flexible deep learning approach not previously applied to this dataset. It further compares VAE-generated features with PCA and hybrid approaches across multiple ML classifiers using consistent experimental settings.

3. Background

The NSL-KDD dataset is a portion of the KDD'99 dataset, which is a subset created to try and solve some of the problems in the original set described in Tavallaee et al. [16]. This dataset is comprised of a selection of data files, but in this report, only two are used: KDDTrain+.txt and KDDTest+.txt. Both files have the same CSV format, with each line representing a packet. Each packet has various attributes, including protocol type, logged in, destination, attempted root login, and number of failed attempts. As well as these attributes, each of them has a designated class. This will either be "normal" (i.e., a benign request) or one of several malicious classes (e.g., rootkit, nmap, and buffer_overflow). However, this paper focuses only on whether the attack is malicious or benign (binary classification). Although NSL-KDD is an older dataset, it remains a widely used benchmark for evaluating IDSs due to its balance between complexity and accessibility. Unlike the original KDD'99 dataset, NSL-KDD addresses several known issues, such as redundant records and class imbalance, making it suitable for consistent comparative analysis. Furthermore, the use of NSL-KDD allows for direct benchmarking against a broad body of existing work, especially studies involving autoencoders and classical ML classifiers. Although more recent datasets like UNSW-NB15 provide additional realism, NSL-KDD was chosen to maintain alignment with prior research and to specifically evaluate whether convolutional VAEs can yield measurable improvements within this well-established testbed.

3.1. PCA

PCA is a method used to combine features in a way that maximizes the variance of each feature. The data points are projected into an n-dimensional space, and the direction of greatest variance is identified. This direction becomes the first new feature for the dataset.

Table 1
Comparison of prior methods on the NSL-KDD dataset

Study	Model/Technique	Reported Accuracy	Pros	Limitations
He et al. [6]	Autoencoder + ML classifiers	Up to 6.74% improvement	Simple architecture and boosts accuracy	No probabilistic latent space; limited exploration of feature types
Al-Qatf et al. [7]	Autoencoder + SVM	84.96% (binary), 80.48% (multiclass)	Fast training and good generalization	Standard AE, no variance control
Ahmad et al. [8]	RNN	83.28% (binary), 81.29% (multiclass)	Captures temporal behavior	Computationally intensive; complex tuning
Xu et al. [9]	Deep AE (five layers) + Anomaly detection	90.61%	High accuracy and good visualizations	Error-based detection only; not classifier-based
Song et al. [10]	AE with variable size + IOT extension	Varies	Scalable to IOT and adaptive	Focus not solely on NSL-KDD; model generalization not tested across classifiers

The space is then adjusted to remove the variance associated with this calculated direction. The process is repeated for the updated vector space, identifying the next direction of highest variance to define the second feature. This cycle continues until all dimensions are exhausted, with the number of new dimensions matching the original number of features. It is important to note that PCA neither adds nor removes data but reorganizes the features based on variance. Figure 1 illustrates a PCA transformation on a 2D dataset, where the x- and y-axes are transformed into new axes, x′ and y′. The axes remain perpendicular, preserving the independence between features and ensuring no information is lost.

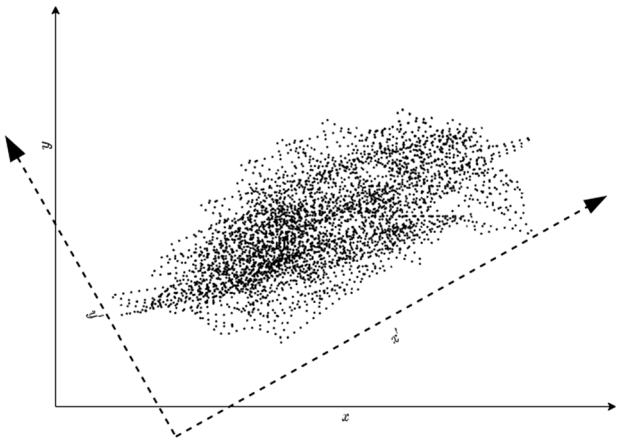
3.1.1. Variance threshold

PCA by itself does not reduce the number of features. To decrease the number of features, a threshold can be established to eliminate all remaining features once a specified percentage of variance is achieved. For example, say after PCA has been performed on a dataset, Feature 1 has 50% of the variance of the dataset, Feature 2 has 26%, and Features 3 and 4 have 4% and 2%, respectively. With a threshold of 75%, Features 1 and 2 would be retained, and Features 3 and 4 would be discarded. This percentage can be found by dividing the specific features' variance by the total summed variance. Typically, this percentage threshold is set to 99% [17].

3.1.2. Maximum likelihood estimate

With variance thresholding, the percentage cutoff is arbitrary. This is still often enough, but the maximum likelihood estimate (MLE) provides a method to select the optimum number of features. The Automatic Choice of Dimensionality for PCA [18] presents a method that employs Bayesian model selection to identify the true dimensionality of the data.

Figure 1
Example of a PCA transformation



3.2. Autoencoders

Autoencoders represent a category of deep NNs that differ from typical NNs in their purpose and structure. Although traditional NNs aim to classify or generate insights from input data, autoencoders strive to replicate the input data as accurately as possible. To achieve this, autoencoders leverage the alteration of node numbers across layers, which allows them to generate intriguing latent features within the network. One crucial component of autoencoders is the bottleneck layer, which contains fewer neurons compared to the other layers. This deliberate restriction of available features forces the network to generate meaningful latent features within this compressed layer. As shown in Figure 2, an autoencoder comprises two main parts: the encoder, responsible for DR, and the decoder, which focuses on data reconstruction. These two parts are connected by each of their aforementioned bottleneck layers [19]. Autoencoder training involves arranging the encoder and decoder parts in series and minimizing the discrepancy between the input and output data. The loss function measures the difference between x and f (g(x)), where x is a data entity, f is the decoder function, and g is the encoder function (e.g., the mean squared error). The latent features are also denoted as z, but it is important to note that z = g(x).

VAEs, shown in Figure 3, offer a probabilistic extension to the standard autoencoder model. Like traditional autoencoders, VAEs aim to reconstruct input data points. However, at the bottleneck layer, instead of passing the exact latent feature values, a probability distribution is created around those values, and a sample from this distribution is passed to the decoder. This sampling method assumes that nearby feature values should produce similar results, enhancing the model's robustness and generalization. Due to this probabilistic nature, the loss function employed in VAEs differs from that of traditional autoencoders. Rather than comparing the difference between two specific values, it focuses on the distinction between two distributions. To accomplish this, the evidence lower bound function is utilized [20].

$$\log p(x) \geq \mathrm{ELBO} = \mathrm{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(Z|X)}\right] \quad (1)$$

Given that x represents the input value, q denotes the encoder, p represents the decoder, and z signifies the latent feature distribution,

Figure 2
An autoencoder with five input features and two latent dimension features

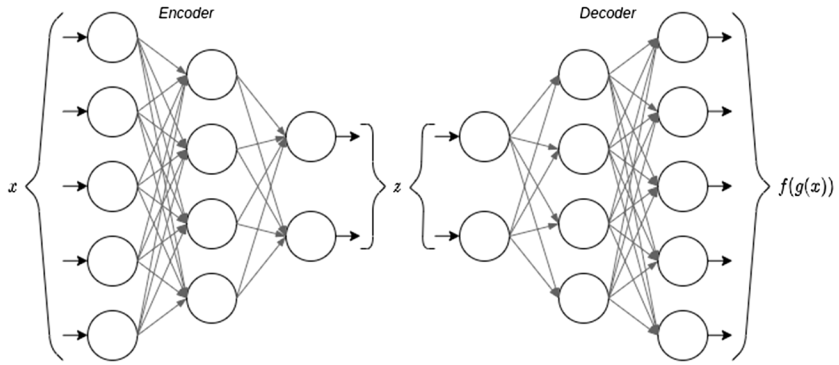
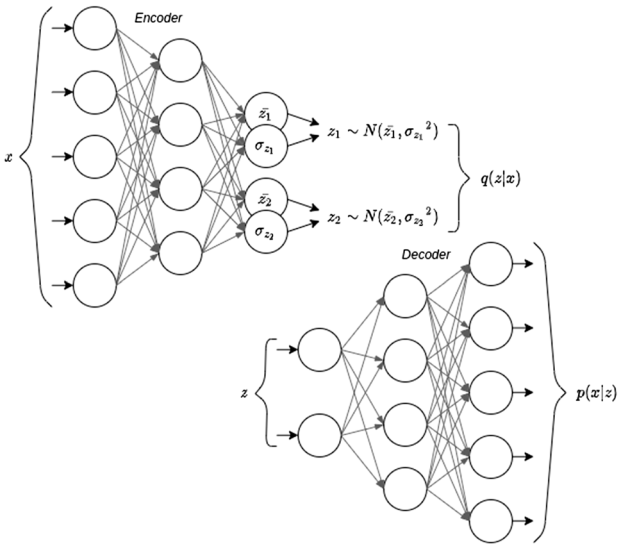


Figure 3
A VAE with five input variables and two latent dimensions



we incorporate the distribution into the decoder by conducting a Monte Carlo simulation on z. This simulation involves passing z through the decoder and subsequently reconstructing the distribution. The Monte Carlo estimate of this expectation is calculated as follows:

$$\approx \log p(x|z) + \log z - \log q(z|x) \tag{2}$$

This loss function algorithm is adapted from TensorFlow [21]. In the encoder, the number of output nodes is double that of the decoder. The encoder nodes are organized into pairs, with one node indicating the mean of a distribution and the other indicating the standard deviation. This technique is known as "reparameterization."

$$z = \mu + \sigma \odot \epsilon \tag{3}$$

where ϵ denotes an error term that follows a standard normal distribution.

A key aspect that distinguishes VAEs from traditional autoencoders and other DR techniques lies in their probabilistic framework. Unlike standard autoencoders, which map inputs to a deterministic latent space, VAEs encode inputs as probability distributions, specifically, Gaussian distributions characterized by a mean and variance. This probabilistic structure enforces smoothness and continuity in the latent space, which improves generalization and robustness to noise, making VAEs particularly well-suited for applications like anomaly detection and intrusion classification. In contrast, PCA performs linear transformations and cannot model nonlinear relationships or generate new data instances. Standard AEs may capture complex patterns, but they lack the regularization effect and generative capacity that come from sampling in VAEs. Furthermore, the reparameterization trick used in VAEs allows gradients to flow through stochastic nodes, enabling end-to-end training using backpropagation. These properties make VAEs a compelling choice for feature learning in cybersecurity, where the goal is to discover compressed, abstract representations that retain the underlying structure of malicious versus benign behavior.

4. Methodology and Experimentation

The aim of this experiment is to investigate how altering sample features influences the performance of ML classifiers. The experiment is structured into several key phases, as shown in Figure 4, as follows.

4.1. Dataset preprocessing

The NSL-KDD dataset, composed of the KDDTrain+ and KDDTest+ files, contains 41 features describing individual network connections, including three categorical attributes (protocol_type, service, and flag) and 38 numerical attributes such as duration, src_bytes, dst_bytes, wrong_fragment, num_failed_logins, count, and srv_count. These features were retained in full, and no synthetic or engineered features were added, in order to maintain consistency across all DR techniques applied. Each feature captures behavior known to be relevant to network intrusion detection, for example, num_failed_logins indicates brute-force attempts, while count and srv_count reflect potential scanning or flooding behaviors. A preliminary inspection confirmed that the dataset contains no missing values. To reduce the influence of extreme numerical values without distorting the dataset structure, outlier clipping was applied using a z-score method: values beyond three standard deviations from the mean were clipped at the 99th percentile. Importantly, no data points were removed; this nondestructive approach preserves the original dataset while mitigating the impact of skewed distributions, particularly for models sensitive to feature scale. Although no ablation study was conducted to directly compare clipped versus unclipped performance, this remains a potential avenue for future research. Categorical features were transformed using one-hot encoding, expanding the total feature count and allowing compatibility with NNs and SVMs. Numerical features were scaled to the [0,1] range via min-max normalization to stabilize training, particularly for autoencoders. The original multi-class labels were converted into a binary classification task, where "0" denotes benign traffic and "1" denotes any type of malicious activity. Class imbalance, while present, was addressed through stratified sampling during train-test splitting to maintain proportional representation. These preprocessing choices

Figure 4
Experimentation phases

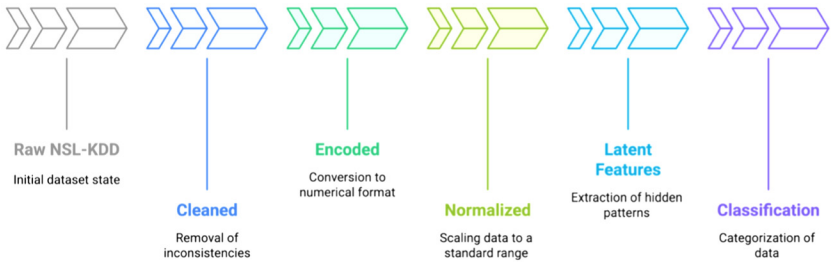
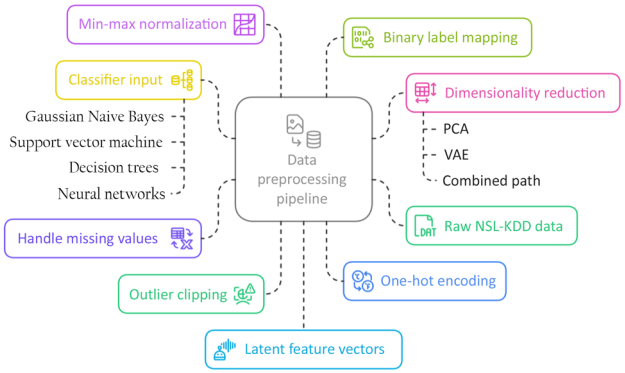


Figure 5
Overview of the data transformation pipeline from raw NSL-KDD records to processed feature vectors used in training and evaluation



Note: PCA = Data Preprocessing Pipeline, VAE = variational autoencoder.

ensure model compatibility, enhance numerical stability, and preserve the predictive value and interpretability of features. To clarify how the raw dataset was transformed into model-ready inputs, Figure 5 presents a high-level overview of the data transformation pipeline. The process begins with the raw NSL-KDD records, followed by missing value checks and outlier clipping. Categorical features are then one-hot encoded, and numerical attributes are scaled. Labels are binarized, after which the data may follow one of three DR paths: PCA only, VAE only, or a hybrid PCA→VAE pipeline. Each path produces a latent feature representation used for model training. This transformation overview connects each preprocessing step to the broader experimental framework and highlights how the dataset evolves through each stage.

4.2. DR

At this stage, the chosen DR methods are applied, each with its own set of unique parameters. By adjusting these parameters, we can better assess the performance of each method. The following techniques shown in Table 2 are implemented on the raw datasets and then prepared for further processing:

Table 2
DR methods

Method	Description
None	The data is unmodified to act as a control.
PCA Variance Threshold [17]	PCA is applied to the dataset, and then a percentage variance threshold is applied. This threshold is varied between 90% and 99%.
PCA MLE Threshold [22]	PCA is applied to the dataset, and then an MLE threshold is applied.
Variational Autoencoder [23, 24]	A variational encoder is trained on the dataset, and then from that, the latent features are extracted. The number of epochs is varied from 100 to 500, and the number of latent features is also varied from 2–14.
Combinations	Combinations of PCA and autoencoders are also constructed. For each PCA type, a set of autoencoders is trained with the same parameter variances for each.

Note: DR = dimensionality reduction, MLE = maximum likelihood estimate, PCA = "Data Preprocessing Pipeline.

VAEs were selected for this study due to their ability to generate low-dimensional, information-rich representations of high-dimensional data through a probabilistic latent space. Unlike traditional autoencoders or PCA, which rely on deterministic mappings, VAEs model the underlying data distribution by learning a mean and variance for each latent dimension. This enables the generation of smoother, more generalized feature spaces that are robust to noise and overfitting properties, particularly beneficial in cybersecurity applications, where attack patterns may vary widely and unpredictably. Prior research in image processing and anomaly detection has shown that VAEs outperform basic autoencoders in capturing meaningful structure within data [23, 24], and this study explores whether such benefits translate effectively to network intrusion detection. A convolutional structure was adopted to allow the encoder to exploit localized patterns in the input feature space, similar to how spatial correlations are modeled in image data. Additionally, the number of latent dimensions (2–14) and training epochs (100 to 500) were varied to empirically determine the optimal configuration for downstream classification. These hyperparameter ranges were chosen based on common practices in deep learning research and validated by observing performance trends across multiple models and metrics.

4.3. Model creation/training

The following models were trained on datasets processed using various DR techniques: VAEs, PCA, and combinations thereof. The models include GNB Networks, SVMs with an RBF kernel, decision trees, and dense NNs. Each model was evaluated on its performance and generalization ability using accuracy, precision, recall, F1 score, and the Matthews Correlation Coefficient (MCC), as detailed in the performance metrics summarized in Table 3.

4.3.1. Decision trees

For decision trees, the following hyperparameters were varied to control model complexity and mitigate overfitting:

1) criterion: Gini impurity or entropy
2) splitter: Best or random
3) min_samples_split: 2, 4, 6, or 8
4) min_samples_leaf: 1 to 4 (inclusive)

These settings were adjusted to prevent overly deep trees and ensure sufficient samples at each split, thereby acting as a form of built-in pruning and regularization. By increasing min_samples_split and min_samples_leaf, the model avoids tailoring its branches too specifically to noise in the training set, which enhances generalizability. A full summary of model-specific training configurations is provided in Table 4.

Table 3
Model training parameters

Model type	Parameters
Gaussian Naïve Bayesian Networks [25]	None
SVM [26, 27]	An RBF kernel is used for the SVM
Decision trees [28]	criterion: gini or entropy splitter: best or random min_samples_split: 2, 4, 6, or 8 min_samples_leaf: between 1 and 4 inclusive

Note: RBF = Radial Basis Function, SVM = support vector machine.

Table 4
Mean improvement for each DR technique compared to the baseline metric for Gaussian Naïve Bayes networks

DR type	VAE	PCA	PCA + VAE
Accuracy	0.271496	0.280635	0.290982
Precision	0.324512	0.364020	0.366039
Recall	−0.065930	−0.095875	−0.069866
F1	0.128701	0.127246	0.141484
MCC	0.692453	0.726935	0.743009

Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, PCA = "Data Preprocessing Pipeline, VAE = variational autoencoder.

Across all models, overfitting was further controlled by using stratified train-test splits, ensuring balanced representation of classes. Although k-fold cross-validation and ensemble techniques were not implemented in this study, their inclusion in future work could provide additional evidence of model robustness. Performance was evaluated using multiple metrics beyond accuracy, such as F1 and MCC, to better assess the ability to generalize, especially under class imbalance. Autoencoder training was monitored across increasing epochs and latent feature sizes, and only configurations showing consistent test performance gains were retained for evaluation.

4.4. Step-by-step problem-solving flow of the proposed method

Based on our previous work in Taylor and Eleyan [29], the proposed method was designed to address key limitations in IDSs, including high feature dimensionality, weak generalization of traditional models, and inconsistent classifier performance. The following step-by-step approach outlines how the method solves these challenges:

1) Challenge: high-dimensional network data includes redundant and noisy features that degrade classifier performance.

Step 1: feature preprocessing

The NSL-KDD dataset is cleaned and normalized. Categorical features are one-hot encoded, and numerical features are scaled to [0,1]. Outlier clipping is applied to reduce the impact of extreme values. These steps ensure that all inputs are in a consistent and stable format for further processing.

2) Challenge: linear DR methods like PCA cannot capture non-linear attack patterns.

Step 2: DR

The dataset is passed through either PCA, VAE, or a combined PCA→VAE pipeline. PCA removes linear redundancy, while VAEs create probabilistic latent feature representations that capture non-linear, higher-order relationships in the data.

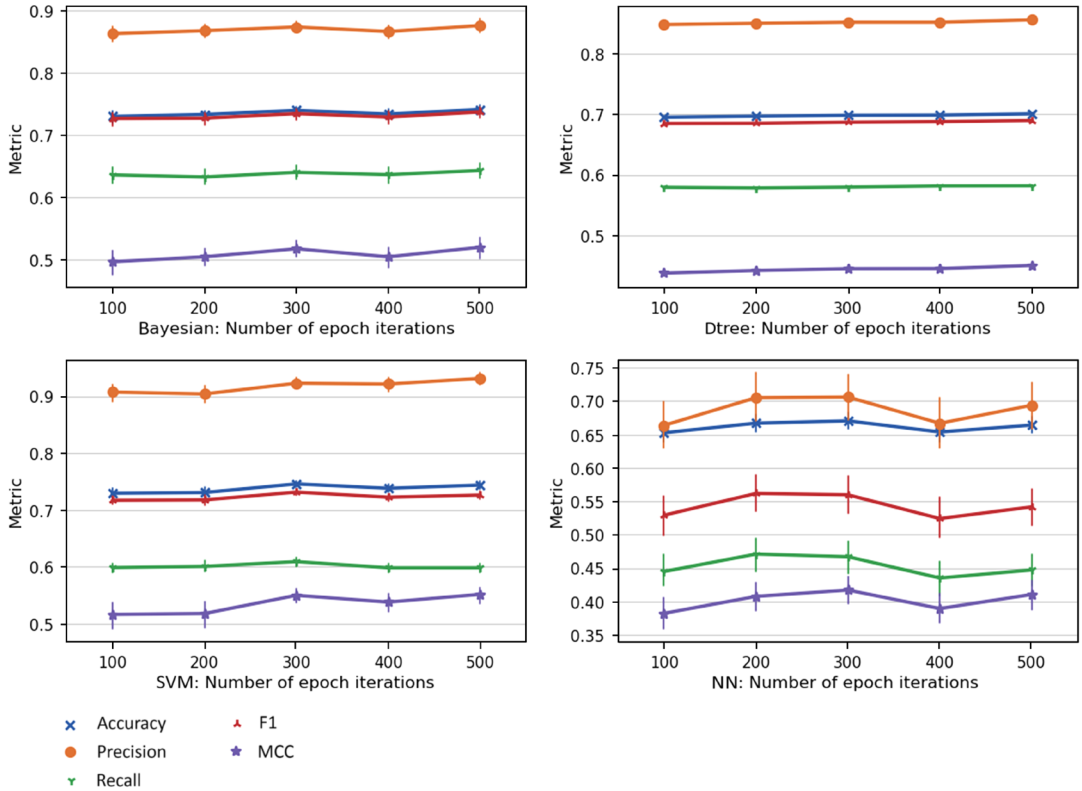
3) Challenge: standard ML models perform inconsistently across raw and reduced feature spaces.

Step 3: hybrid feature engineering

A hybrid configuration (PCA followed by VAE) is proposed to combine the benefits of variance-based filtering with deep, abstract feature generation. This reduces noise while preserving critical patterns, improving the robustness of downstream models.

4) Challenge: classifiers overfit to training data or underperform due to irrelevant features.

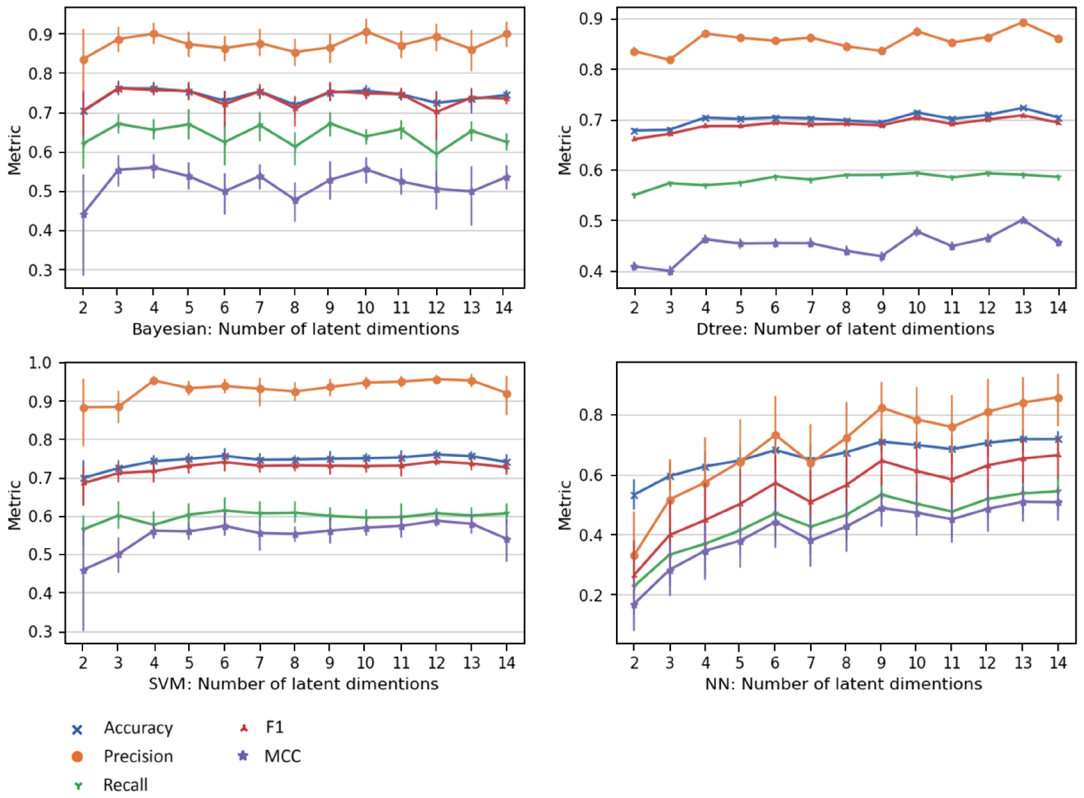
Figure 6
Mean metric value across ALL DR combinations with an autoencoder against the number of training epochs



Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, NN = neural network, SVM = support vector machine.

Figure 7

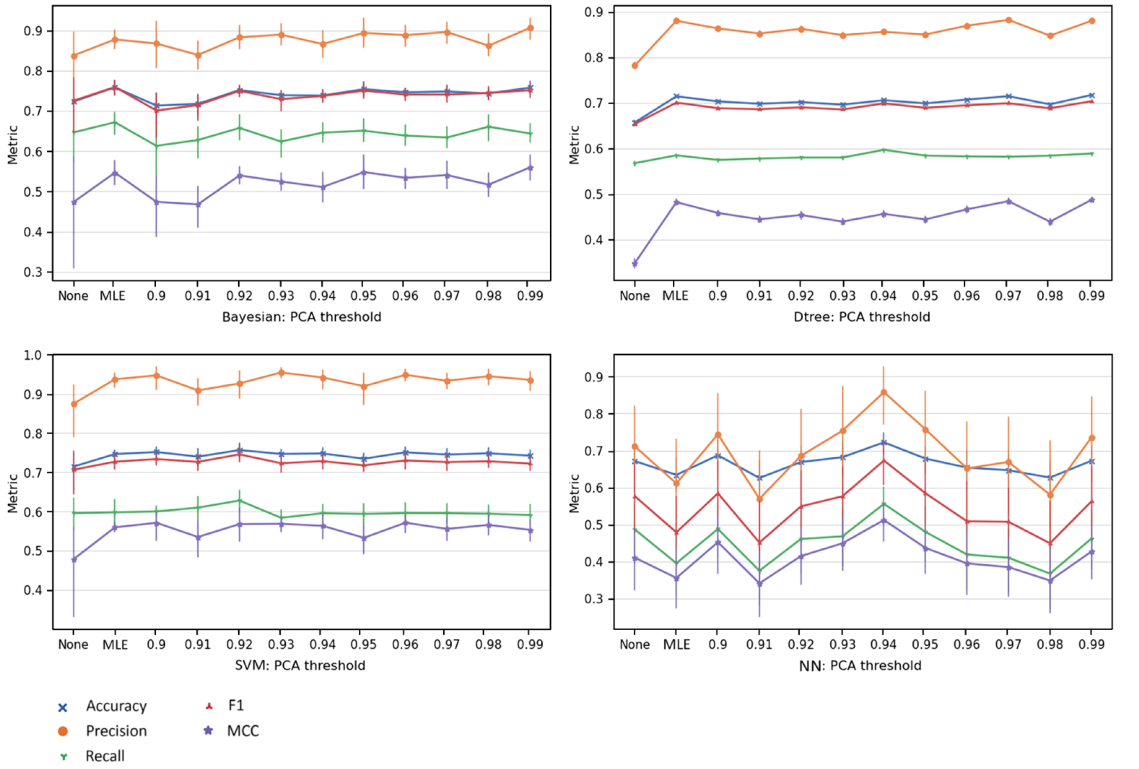
Mean metric values across ALL DR combinations using an autoencoder versus the number of latent feature dimensions



Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, NN = neural network, SVM = support vector machine.

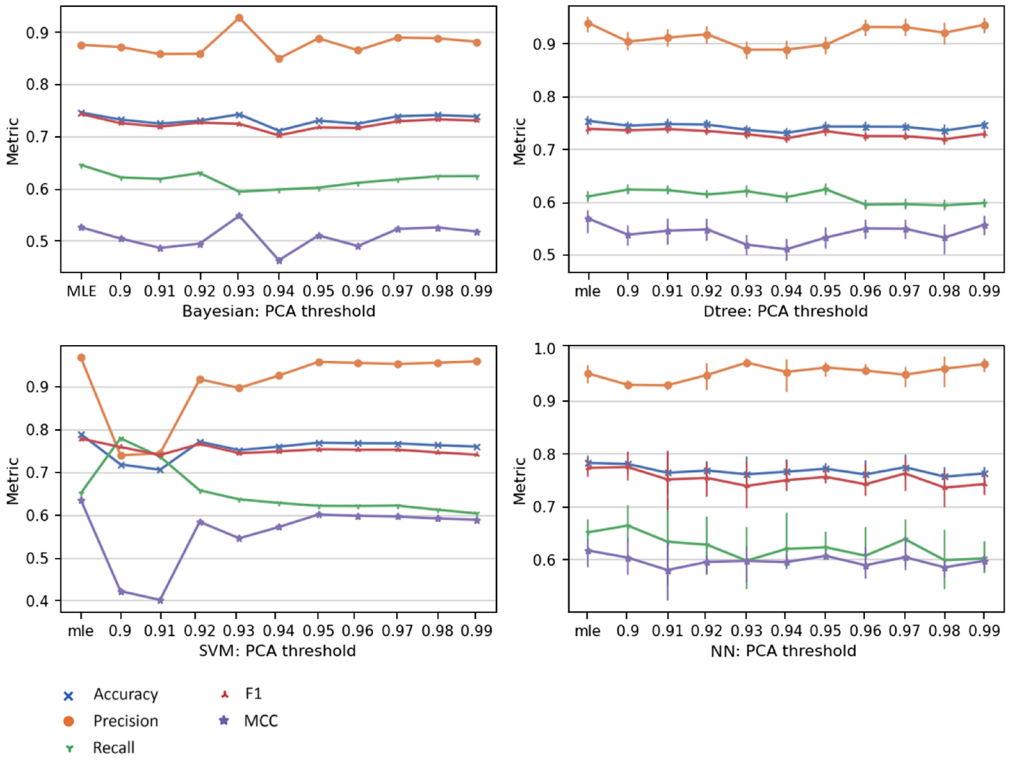
Figure 8

Mean metric value across ALL DR combinations with an autoencoder against the type of PCA variance threshold



Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, MLE = maximum likelihood estimate, NN = neural network, PCA = Data Preprocessing Pipeline, SVM = support vector machine.

Figure 9
Mean metric value across all combinations of ML classifiers for each PCA variance threshold type



Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, ML - machine learning, MLE = maximum likelihood estimate, NN = neural network, PCA = Data Preprocessing Pipeline, SVM = support vector machine.

Step 4: classifier training

Multiple classifiers (GNB, SVM-RBF, decision trees, NNs) are trained on these transformed features. Hyperparameters are tuned to balance complexity and generalization. Decision tree pruning and stratified train-test splits are used to avoid overfitting.

5) Challenge: lack of generalizable results across models and datasets.

Step 5: performance evaluation and comparison

Classifier outputs are evaluated using multiple metrics: accuracy, precision, recall, F1 score, and MCC to capture performance comprehensively. The PCA+VAE feature set consistently yields the best results, showing its ability to generalize across models.

6) Challenge: limited insight into which methods are most effective for IDS.

Step 6: result analysis and insights

Comparative results show which classifier-DR combinations perform best. For example, SVM with PCA+VAE achieved 82.8% accuracy and an MCC of 0.682, the highest among all configurations. These findings offer practical guidance for real-world IDS design.

5. Results and Discussion

5.1. DR tuning

To understand the impact of different DR strategies on classification performance, a series of experiments were conducted by varying key parameters of both PCA and VAEs. Specifically, PCA variance thresholds ranging from 90% to 99%, and VAE configurations including training epochs (from 100–500 as shown in Figure 6) and latent dimension sizes (from 2–14 as shown in Figure 7) were tested.

The results reveal that increasing the number of latent features in VAEs generally improves classification metrics across models, though with diminishing returns beyond a certain point (Figure 7). The average metric values improve as more expressive latent representations are learned, indicating that the bottleneck in the VAE is critical for retaining meaningful structure. Similarly, Figure 8 shows that combining PCA with VAE generally leads to better average performance than either method alone, suggesting that PCA may help remove redundant variance before the VAE encodes higher-order abstractions. However, Figure 9 demonstrates that there is no clear optimal PCA threshold across all classifiers performance depends on both the DR combination and the model architecture.

5.2. Classifier performance with DR

Each classifier was evaluated across multiple DR configurations using key metrics: accuracy, precision, recall, F1 score, and the MCC. Figures 10 and 11 show the distribution of these metrics for GNB and SVM-RBF models, respectively.

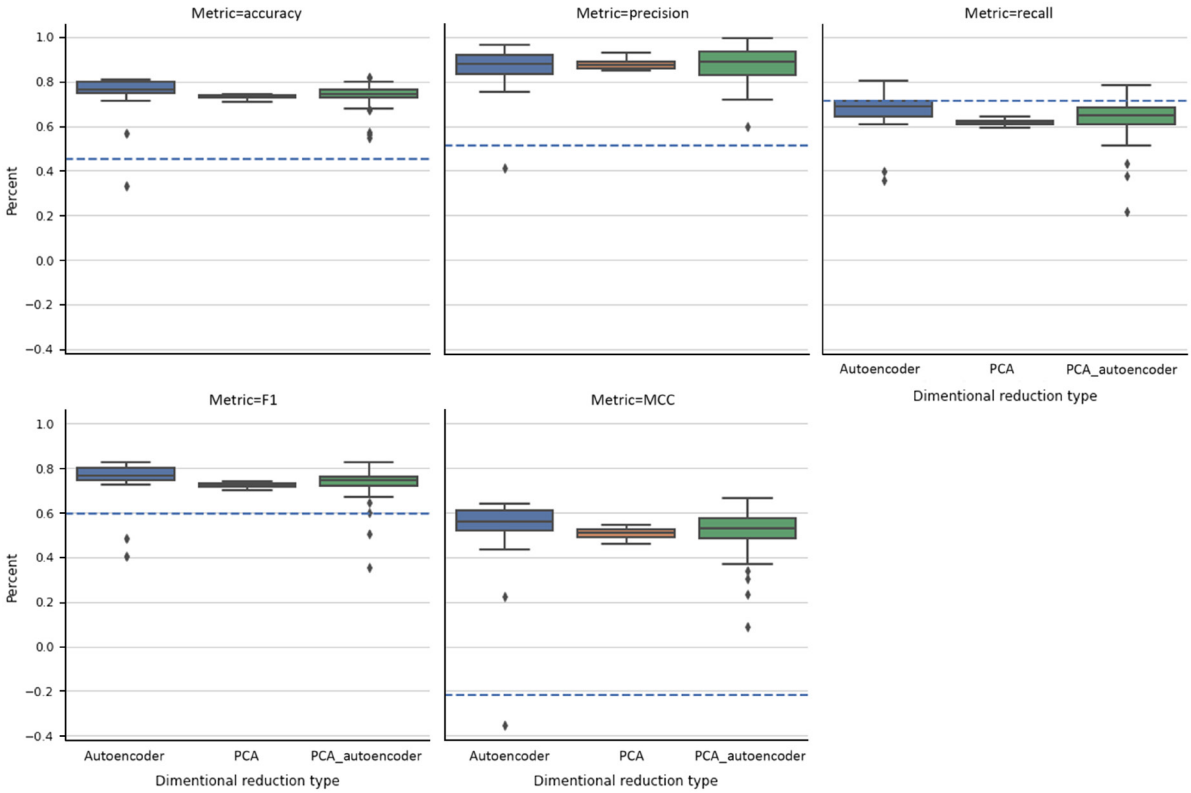
1) GNB

Table 5 summarizes the performance of GNB models under different DR methods. The baseline model (no DR) performed poorly (MCC: −0.219), while applying VAE significantly improved results (MCC: 0.618). PCA also improved performance (MCC: 0.526), but the best result was obtained with a hybrid PCA + VAE configuration (MCC: 0.668). These results suggest that probabilistic latent features generated by VAEs help reduce the naïve assumption violations inherent in GNB.

2) SVMs (SVM-RBF)

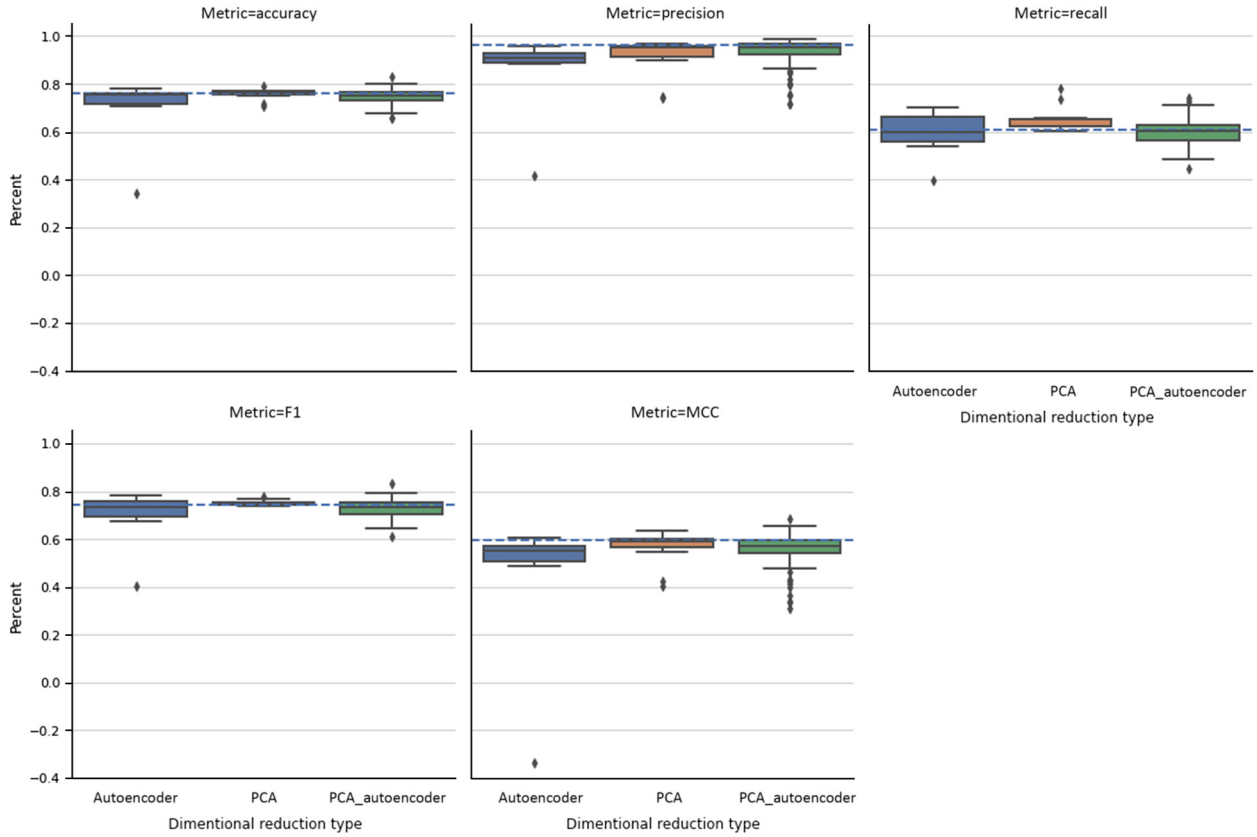
Table 6 presents the mean improvement of each DR method for SVMs. While PCA slightly improved performance, VAE alone slightly reduced average results. Nonetheless, the best-performing SVM model overall used both PCA (threshold 0.92) and a VAE with six latent

Figure 10
Boxplot of all metric types for GNB, grouped by DR method



Note: DR = dimensionality reduction, GNB = Gaussian Naïve Bayes, MCC = Matthews Correlation Coefficient, PCA = Data Preprocessing Pipeline.

Figure 11
Boxplot of all metrics for the SVM with RBF kernel, grouped by DR method



Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, PCA = Data Preprocessing Pipeline, RBF = Radial Basis Function, SVM = support vector machine.

Table 5
Maximum performing GNB network models split by DR method

Meta DR type	DR	Accuracy	Precision	Recall	F1	MCC
None	None	0.451694	0.513249	0.712382	0.596639	-0.219382
VAE	400 epochs, 5 features	0.810282	0.853437	0.804956	0.828488	0.617966
PCA	MLE threshold	0.745697	0.875423	0.645056	0.742788	0.525569
PCA with VAE	0.97 threshold and 500 epochs, 4 features	0.821505	0.937258	0.735681	0.824326	0.667928

Note: DR = dimensionality reduction, GNB = Gaussian Naïve Bayes, MCC = Matthews Correlation Coefficient, MLE = maximum likelihood estimate, PCA = Data Preprocessing Pipeline, VAE = variational autoencoder.

Table 6
Mean metric improvement for each DR method for SVM-RBF models

DR type	VAE	PCA	PCA + VAE
Accuracy	**−0.048422**	**−0.005253**	**−0.016317**
Precision	−0.090248	−0.053932	−0.028641
Recall	−0.009585	0.043702	−0.006658
F1	−0.037584	0.009035	−0.015862
MCC	−0.117931	−0.034707	−0.037293

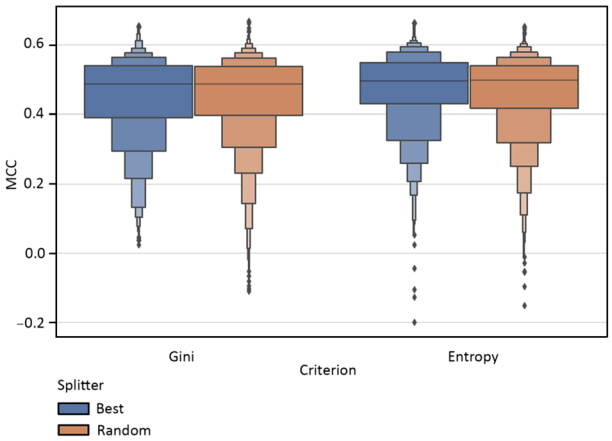
Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, PCA = Data Preprocessing Pipeline, VAE = variational autoencoder.

features (Table 7), achieving an accuracy of 82.8% and MCC of 0.682. This implies that a carefully tuned hybrid DR pipeline can enhance SVM performance, even if average improvements are small.

3) Decision trees

Figures 12 and 13 and Table 8 evaluate decision tree models under different splitting strategies and DR methods. Notably, decision trees performed well even without DR, due to their internal structure that inherently filters irrelevant features. The best-performing decision tree model (gini, random splitter, MSS = 6, MSL = 3) achieved an MCC of 0.667 without DR, closely matching the best DR-enhanced versions. This suggests that decision trees may benefit less from DR than other classifiers due to their embedded feature selection capabilities.

Figure 12
Boxen plot of decision tree MCC split by classification criterion and node splitting strategy



Note: MCC = Matthews Correlation Coefficient.

4) NNs

Figures 14 and 15 show how NN performance varied with DR configurations and model depth. Unlike other models, NNs showed mixed results, with some DR combinations reducing performance on the NSL-KDD dataset. However, Table 9 indicates that certain hybrid configurations, such as PCA@0.91 combined with a VAE (500 epochs, 11 latent features), produced competitive results (accuracy: 0.810, MCC: 0.666). This suggests that DR must be carefully tuned when applied before deep learning models, as unnecessary compression can discard nonlinear relationships critical to NNs.

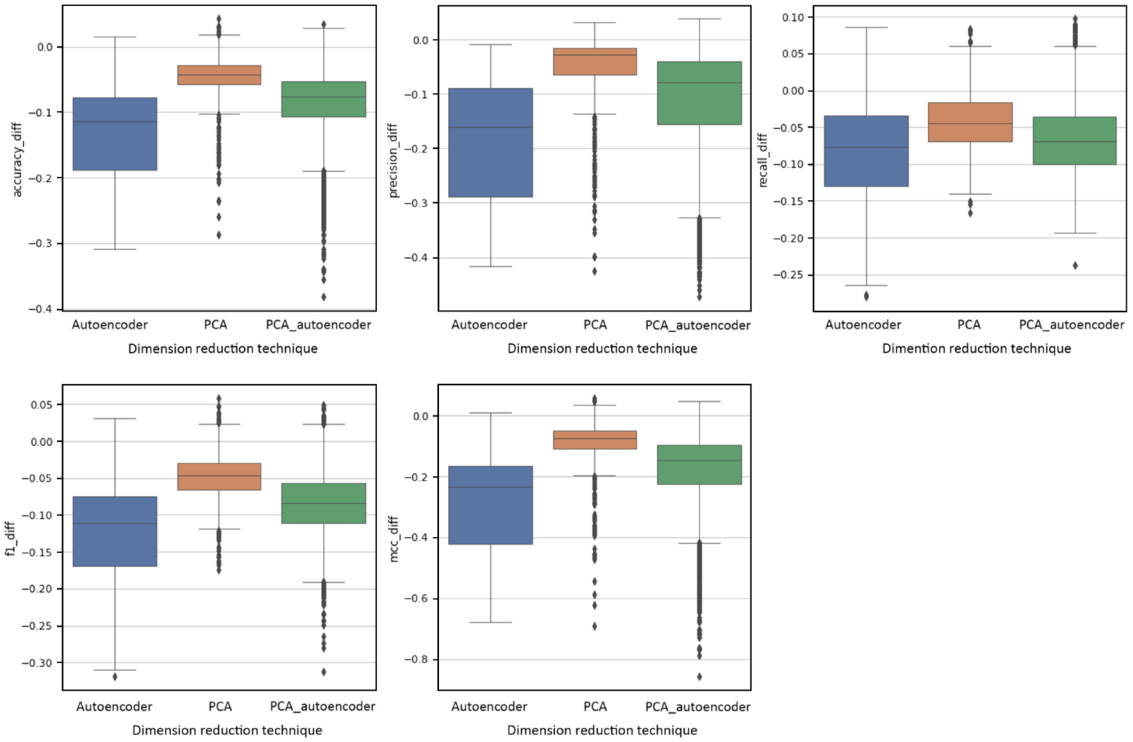
Table 7
Maximum performing SVM-RBF models split by DR method

Meta DR type	DR	Accuracy	Precision	Recall	F1	MCC
None	**None**	**0.762952**	**0.965213**	**0.605392**	**0.744086**	**0.595881**
VAE	500 epochs, 4 features	0.779853	0.933752	0.660095	0.773431	0.604059
PCA	MLE threshold	0.789257	0.968576	0.650900	0.778580	0.634754
PCA with VAE	Threshold 0.92 and 500 epochs, 6 features	0.828070	0.946555	0.739733	0.830461	0.682102

Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, PCA = Data Preprocessing Pipeline, RBF = Radial Basis Function, SVM = support vector machine, VAE = variational autoencoder.

Figure 13

Decision tree metric improvement box plots for each metric split by DR technique



Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, PCA = Data Preprocessing Pipeline, RBF = Radial Basis Function, SVM = support vector machine.

Table 8

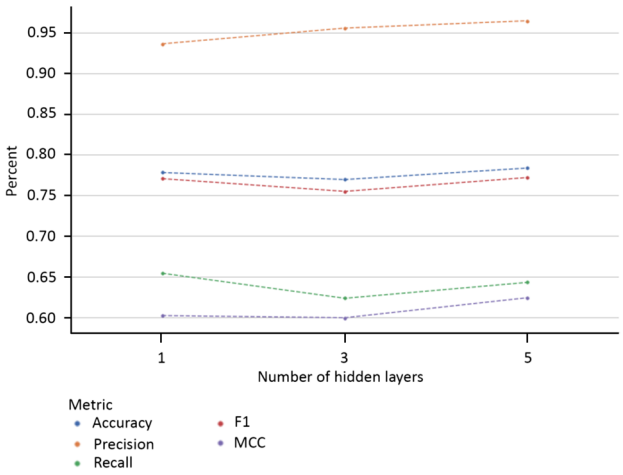
Mean improvement for each DR technique from the base metric for decision trees

Criterion	Splitter	MSS	MSL	DR type	Accuracy	Precision	Recall	F1	MCC
gini	**Best**	**8**	**3**	**none**	**0.801677**	**0.972750**	**0.670381**	**0.793745**	**0.654712**
gini	Random	6	3	none	0.811081	0.970169	0.689317	0.805977	0.667231
gntropy	Best	6	2	none	0.809351	0.967057	0.688537	0.804370	0.663211
gntropy	Random	2	2	none	0.801411	0.968281	0.673186	0.794208	0.652169

Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient.

Figure 14

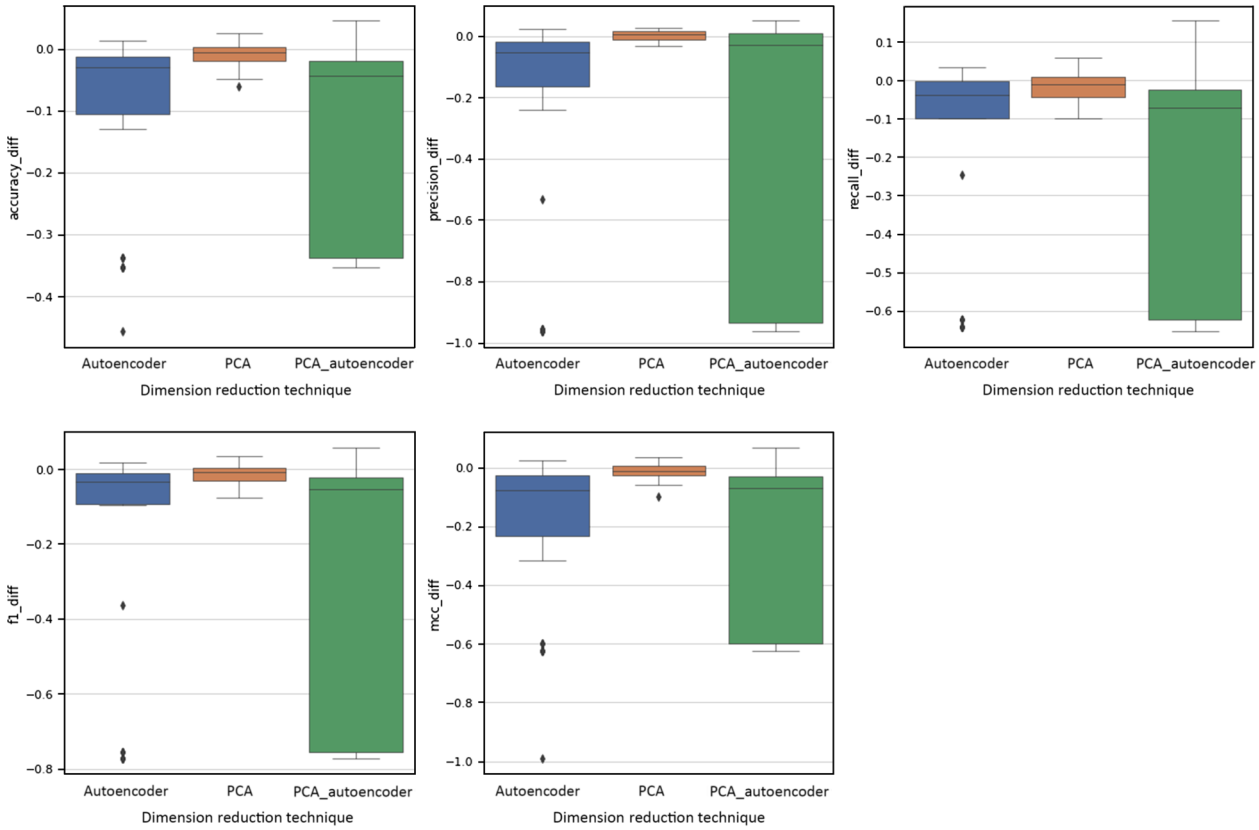
Model performance over model hidden layer depth without DR



Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient.

Table 9 ranks the top 10 performing models across all classifiers and DR methods. The highest overall performance was achieved by the SVM-RBF using PCA@0.92 and VAE (6 latent features), with an accuracy of 82.8% and MCC of 0.682. Interestingly, decision trees with no DR also appeared multiple times among the top models, indicating their robustness even in high-dimensional feature spaces. For GNB, DR significantly improved baseline performance, while for NNs, DR outcomes were highly configuration-dependent. Overall, the results demonstrate that convolutional VAEs can effectively enhance classifier performance when properly configured and, in many cases, outperform both PCA and baseline (no DR) models. The combination of PCA and VAE consistently delivered the best results for GNB and SVM classifiers, while decision trees benefited less due to their inherent feature filtering. NNs showed mixed responses to DR, highlighting the need for tailored configurations depending on the model complexity and feature space. These findings confirm the value of deep probabilistic feature extraction for intrusion detection and emphasize the importance of empirical tuning in DR-classifier pipelines.

Figure 15
Box plot of each metric improvement for NNs split by DR technique



Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, NN = neural network, PCA = Data Preprocessing Pipeline, RBF = Radial Basis Function, SVM = support vector machine.

Table 9
Top 10 performing models

DR type	Model type	Accuracy	Precision	Recall	F1	MCC
None	dtree entropy best mss 8 msl 1	0.806911	0.969027	0.682615	0.800988	0.660562
None	dtree gini random mss 2 msl 2	0.808419	0.966879	0.686979	0.803244	0.661755
None	dtree entropy best mss 4 msl 2	0.808818	0.968554	0.686433	0.803448	0.663137
None	dtree entropy best mss 6 msl 2	0.809351	0.967057	0.688537	0.804370	0.663211
None	dtree gini random mss 4 msl 1	0.809972	0.969880	0.687524	0.804651	0.665462
PCA @ 0.91 VAE 500 ep 11 LFD	nn1	0. 809839	0.971842	0.685810	0.804148	0.666197
none	dtree gini random mss 6 msl 3	0.811081	0.970169	0.689317	0.805977	0.667231
PCA @ 0.9 VAE 500 ep 9 LFD	NN3	0.814674	0.959541	0.704122	0.812225	0.667584
PCA @ 0.97 VAE 500 ep 4 LFD	bayesian	0.821505	0.937258	0.735681	0.824326	0.667928
PCA @ 0.92 VAE 500 ep 6 LFD	svm rbf	0.828070	0.946555	**0.739733**	**0.830461**	0.682102

Note: DR = dimensionality reduction, MCC = Matthews Correlation Coefficient, PCA = Data Preprocessing Pipeline, RBF = Radial Basis Function, SVM = support vector machine, VAE = variational autoencoder.

Table 10
Performance comparison with existing methods

Year	Method	Dataset	Accuracy	F1 score	Notes
2024	Red Fox + RBF [11]	IoT	91.2%	0.89	High adaptability, complex setup.
2024	PPGO + Naïve Bayes [13]	NSL-KDD	86.3%	0.84	Probabilistic, optimized selection.
2024	WI-CS + GNN [14]	SCADA	92.4%	0.91	Graph-based, domain-specific.
2025	Ours (PCA@0.92+VAE-6 + SVM)	NSL-KDD	82.8%	0.830	Strong MCC (0.682), generalizable.

Note: MCC = Matthews Correlation Coefficient, PCA = Data Preprocessing Pipeline, RBF = Radial Basis Function, SVM = support vector machine, VAE = variational autoencoder.

5.3. Comparative study with existing methods

To evaluate the effectiveness of our method in the context of current research, we compare in Table 10 our best-performing model with recent state-of-the-art intrusion detection methods published in 2023 and 2024.

This comparison demonstrates that although some recent models outperform ours in terms of raw accuracy on specialized datasets, our model achieves competitive results on the NSL-KDD benchmark while also offering modularity and extensibility. Unlike domain-specific frameworks or black-box optimizations, our approach emphasizes interpretability and methodological rigor.

5.4. Model interpretability and insights

Although the primary focus of this study was on improving classification performance through DR, model interpretability remains essential for practical deployment, especially in cybersecurity settings where understanding the reasoning behind an alert is critical. For tree-based models such as decision trees, interpretability is inherently high splits based on features like num_failed_logins, src_bytes, or same_srv_rate directly indicate thresholds that trigger malicious classifications. These insights can guide network administrators toward specific traffic patterns or anomalies that may require investigation. In contrast, models such as SVMs and NNs, particularly those using VAE-generated features, are less interpretable by default. However, DR itself offers some level of abstraction: the latent features learned by VAEs often represent compressed patterns across the input space. For example, a VAE may learn to compress high dst_host_srv_count and frequent serror_rate combinations into a single latent feature that correlates strongly with denial-of-service attacks. Though the latent dimensions are not directly human-readable, their contribution to downstream classification can be visualized or analyzed using tools such as SHAP (SHapley Additive exPlanations) or feature attribution heatmaps in future work. Ultimately, understanding which original input features are most influential even after compression can help translate model recommendations into actionable cybersecurity policies. Future extensions of this work could include training explainable models (e.g., XGBoost with SHAP interpretation) or mapping latent dimensions back to feature groupings for enhanced transparency.

5.5. Computational overhead and complexity considerations

Although the integration of DR techniques, particularly VAEs, yielded performance improvements in several ML classifiers, it also introduced additional computational overhead. Training VAEs, especially with deeper architectures and multiple latent dimensions, involves increased training time and memory usage due to the backpropagation of both mean and variance terms, as well as sampling operations in the latent space. Additionally, tuning VAE hyperparameters such as the number of epochs and latent dimensions requires repeated training and evaluation, which can be computationally expensive. Compared to PCA, which is relatively lightweight and deterministic, VAEs demand GPU resources and careful convergence monitoring. The downstream classifiers themselves (e.g., SVMs and NNs) also experienced slight increases in training time due to the larger feature sets produced after one-hot encoding and normalization. Although these overheads were manageable in our experiments using moderate-sized datasets, they could pose scalability challenges for real-time or large-scale IDSs. Future work could explore optimization strategies such as model pruning, feature selection before encoding, or using lightweight VAE variants to reduce the computational footprint while preserving accuracy.

5.6. Security analysis and limitations

This work focused on enhancing ML-based intrusion detection performance using DR techniques such as PCA and VAEs. Although the models were evaluated extensively using performance metrics like accuracy, precision, F1 score, and MCC, this study did not include a formal security analysis using tools such as ProVerif or Scyther. These tools are well-suited for protocol-level security verification and reasoning about cryptographic properties, but are less directly applicable to evaluating data-driven ML systems where behaviors are learned from traffic data rather than defined through protocol rules. Nevertheless, we acknowledge the value of formal analysis in validating the trustworthiness of IDSs, especially in adversarial or high-assurance environments. An informal security evaluation was performed by analyzing how well the models detect various types of malicious traffic (e.g., denial-of-service, brute-force login attempts, and port scans). By assessing false positive and false negative rates under multiple DR scenarios, we ensured the system's ability to generalize and maintain reliable decision boundaries across different attack categories. In future work, we plan to integrate formal verification techniques to assess specific components, such as the robustness of feature extraction against adversarial inputs or the consistency of classification decisions under transformation. Additionally, formal verification frameworks could be employed to ensure traceability and auditability of decisions made by high-impact detection models.

5.7. Limitations and future directions

Although the proposed approach demonstrates improved classifier performance through the use of VAEs and DR, several limitations remain. First, the evaluation was restricted to the NSL-KDD dataset, which, despite being a widely used benchmark, lacks the diversity and real-time variability of modern traffic datasets. As such, the generalizability of the model to newer or more complex environments

(e.g., IoT or cloud networks) remains untested. Second, although VAEs improve performance, they require longer training times and are computationally more expensive than linear methods like PCA. This could hinder deployment in real-time or resource-constrained systems. Third, interpretability is reduced when using deep learning models such as VAEs and NNs, limiting their use in environments where transparency is essential. Additionally, this study did not explore the model's resistance to adversarial attacks or evasion techniques, which are important considerations for operational security systems.

Future work should focus on validating the model across external datasets such as UNSW-NB15 or CICIDS2017, incorporating formal generalizability testing through cross-validation or domain adaptation. Research could also explore hybrid models that balance interpretability and performance and investigate lightweight or adversarially robust variants of VAEs for deployment in real-time IDS. Finally, incorporating explainability tools such as SHAP or LIME could enhance the transparency and auditability of the model's predictions, while integrating NLP models could improve user experience.

6. Conclusion

This study demonstrated that VAEs, particularly when combined with PCA, can significantly enhance the performance of ML classifiers for intrusion detection. Using the NSL-KDD dataset, we showed that models trained on VAE-generated latent features consistently outperformed those trained on raw or PCA-only features across several evaluation metrics. The best-performing configuration, an SVM with RBF kernel using a hybrid PCA-VAE feature set, achieved an accuracy of 82.8% and an MCC of 0.682, highlighting the value of deep feature representation in cybersecurity contexts. By integrating VAEs into the DR pipeline, this work contributes a flexible and effective approach for improving the accuracy and robustness of IDSs. Although the current evaluation focused on a controlled dataset, this methodology lays a strong foundation for future research into scalable, interpretable, and generalizable IDS solutions.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in NSL-KDD Dataset at https://github.com/t-taylor/cvaes-research/tree/main/results.

Author Contribution Statement

Thomas Taylor: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Amna Eleyan:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Mohammed Al-Khalidi:** Validation, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision.

References

[1] Ali, T., Eleyan, A., Bejaoui, T., & Al-Khalidi, M. (2024). Lightweight intrusion detection system with GAN-based knowledge distillation. In *2024 International Conference on Smart Applications, Communications and Networking*, 1–7. https://doi.org/10.1109/SmartNets61466.2024.10577682

[2] Ali, T., Al-Khalidi, M., Al-Zaidi, R., Eleyan, A., & Rehman, M. A. U. (2024). Securing the metaverse: A deep reinforcement learning and generative adversarial network approach to intrusion detection. In *2024 IEEE International Conference on Communications Workshops*, 263–268. https://doi.org/10.1109/ICCWorkshops59551.2024.10615630

[3] Al-Khalidi, M., Al-Zaidi, R., Ali, T., Khan, S., & Bashir, A. K. (2025). AI-optimized elliptic curve with certificate-less digital signature for zero trust maritime security. *Ad Hoc Networks*, *166*, 103669. https://doi.org/10.1016/j.adhoc.2024.103669

[4] Haq, B., Jamshed, M. A., Ali, K., Kasi, B., Arshad, S., Kasi, M. K., ..., & Ur-Rehman, M. (2024). Tech-driven forest conservation: Combating deforestation with internet of things, artificial intelligence, and remote sensing. *IEEE Internet of Things Journal*, *11*(14), 24551–24568. https://doi.org/10.1109/JIOT.2024.3378671

[5] Markus, A. F., Kors, J. A., & Rijnbeek, P. R. (2021). The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies. *Journal of Biomedical Informatics*, *113*, 103655. https://doi.org/10.1016/j.jbi.2020.103655

[6] He, K., Kim, D. D., & Asghar, M. R. (2023). Adversarial machine learning for network intrusion detection systems: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *25*(1), 538–566. https://doi.org/10.1109/COMST.2022.3233793

[7] Al-Qatf, M., Lasheng, Y., Al-Habib, M., & Al-Sabahi, K. (2018). Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. *IEEE Access*, *6*, 52843–52856. https://doi.org/10.1109/ACCESS.2018.2869577

[8] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, *32*(1), e4150. https://doi.org/10.1002/ett.4150

[9] Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y., & Sabrina, F. (2021). Improving performance of autoencoder-based network anomaly detection on NSL-KDD dataset. *IEEE Access*, *9*, 140136–140146. https://doi.org/10.1109/ACCESS.2021.3116612

[10] Song, Y., Hyun, S., & Cheong, Y. G. (2021). Analysis of autoencoders for network intrusion detection. *Sensors*, *21*(13), 4294. https://doi.org/10.3390/s21134294

[11] Rabie, O. B. J., Selvarajan, S., Hasanin, T., Alshareef, A. M., Yogesh, C. K., & Uddin, M. (2024). A novel IoT intrusion detection framework using Decisive Red Fox optimization and descriptive back propagated radial basis function models. *Scientific Reports*, *14*(1), 386. https://doi.org/10.1038/s41598-024-51154-z

[12] Prashanth, S. K., Shitharth, S., Praveen Kumar, B., Subedha, V., & Sangeetha, K. (2022). Optimal feature selection based on evolutionary algorithm for intrusion detection. *SN Computer Science*, *3*(6), 439. https://doi.org/10.1007/s42979-022-01325-4

[13] Shitharth, S., Kshirsagar, P. R., Balachandran, P. K., Alyoubi, K. H., & Khadidos, A. O. (2022). An innovative perceptual pigeon galvanized optimization (PPGO) based Likelihood Naïve Bayes (LNB) classification approach for network intrusion detection

system. *IEEE Access*, 10, 46424–46441. https://doi.org/10.1109/ACCESS.2022.3171660

[14] Shitharth, S., Satheesh, N., Kumar, B. P., & Sangeetha, K. (2021). IDS detection based on optimization based on WI-CS and GNN algorithm in SCADA network. In S. K. Das, S. Samanta, N. Dey, B. S. Patel, & A. E. Hassanien (Eds.), *Architectural wireless networks solutions and security issues* (pp. 247–265). *Springer.* https://doi.org/10.1007/978-981-16-0386-0_14

[15] Mummadi , A., Yadav, B. M. K., Sadhwika, R., & Shitharth, S. (2022). An appraisal of cyber-attacks and countermeasures using machine learning algorithms. In *International Conference on Artificial Intelligence and Data Science*, 27–40. https://doi.org/10.1007/978-3-031-21385-4_3

[16] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6. https://doi.org/10.1109/CISDA.2009.5356528

[17] Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. https://doi.org/10.1080/14786440109462720

[18] Minka, T. (2000). Automatic choice of dimensionality for PCA. In *Proceedings of the 14th International Conference on Neural Information Processing Systems*, 577–583.

[19] Liou, C. Y., Cheng, W. C., Liou, J. W., & Liou, D. R. (2014). Autoencoder for words. *Neurocomputing*, *139*, 84–96. https://doi.org/10.1016/j.neucom.2013.09.055

[20] Liu, G., van Huynh, N., Du, H., Hoang, D. T., Niyato, D., Zhu, K., ..., & Kim, D. I. (2024). Generative AI for unmanned vehicle swarms: Challenges, applications and opportunities. *arXiv Preprint: 2402.18062*. https://doi.org/10.48550/arXiv.2402.18062

[21] TensorFlow. (2021) *Convolutional variational autoencoder*. https://www.tensorflow.org/tutorials/generative/cvae

[22] Schuermans, M., Markovsky, I., Wentzell, P. D., & Van Huffel, S. (2005). On the equivalence between total least squares and maximum likelihood PCA. *Analytica Chimica Acta*, *544*(1–2), 254–267. https://doi.org/10.1016/j.aca.2004.12.059

[23] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes. *arXiv Preprint: 1312.6114*. https://doi.org/10.48550/arXiv.1312.6114

[24] Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, *32*(2), 1278–1286.

[25] Chan, T. F., Golub, G. H., & LeVeque, R. J. (1982). Updating formulae and a pairwise algorithm for computing sample variances. In *COMPSTAT 1982 5th symposium held at Toulouse 1982: Part I: Proceedings in Computational Statistics*, 30–41. https://doi.org/10.1007/978-3-642-51461-6_3

[26] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

[27] Min, B., Ross, H., Sulem, E., Pouran Ben Veyseh, A., Nguyen, T. H., Sainz, O., ..., & Roth, D. (2023). Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, *56*(2), 30. https://doi.org/10.1145/3605943

[28] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., ..., & Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, *14*(1), 1–37. https://doi.org/10.1007/s10115-007-0114-2

[29] Taylor, T., & Eleyan, A. (2021). Using variational autoencoders to increase the performance of malware classification. In *2021 International Symposium on Networks, Computers and Communications*, 1–6. https://doi.org/10.1109/ISNCC52172.2021.9615643

How to Cite: Taylor, T., Eleyan, A., & Al-Khalidi, M. (2025). Using Variational Autoencoders with Machine Learning Algorithms in Cyber Security Applications. *Artificial Intelligence and Applications*, *3*(4), 428–442. https://doi.org/10.47852/bonviewAIA52024151