A CNN-Autoencoder Framework with 2D Data Transformation and Top-K Reconstruction Scoring for Unsupervised Cybersecurity Anomaly Detection

Hari Krishna Puripanda[1]* & Dr Vemuru Srikanth[2]

Department of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation (Deemed to be University)
Green Fields, Vaddeswaram, Guntur District, Andhra Pradesh, India
haripuripanda2026@gmail.com, vsrikanth@kluniversity.in

Abstract
In this paper, a fully unsupervised anomaly detection framework is presented using a Convolutional Neural Network (CNN) autoencoder to be applied to behavioural security analytics for various heterogeneous security data. The main contribution is to develop a 2D data formulation strategy to convert sequential and tabular security data (such as system logs and network flow data) into fixed-size matrices that enables CNN-based spatial feature extraction for the 2D data types. Two additional works focus on key shortcomings of current autoencoder-based detection: a top-K reconstruction error scoring function that retains the anomaly signals in sparse feature deviations and an F1 optimal threshold selection method employing a held-out calibration set. There is a severity-weighted prioritization module to prioritize flagged anomalies for analysts' triage. It is trained solely based on normal data, tested on NSL-KDD, UNSW-NB15 and CICIDS2017, and obtains the F1 score of 85.83% and 66.86% on the two detection benchmarks in the absence of attack data. Inference can run at up to 143,767 samples per second, with a latency time per sample of 0.007 ms, thus ensuring real-time enterprise deployment.

Keywords: Anomaly Detection, Convolutional Neural Network, Autoencoder, Cybersecurity, Data Representation, Unsupervised Learning, Threat Prioritization

How to cite this article: Puripanda HK, Srikanth V., A CNN-Autoencoder Framework with 2D Data Transformation and Top-K Reconstruction Scoring for Unsupervised Cybersecurity Anomaly Detection. Int J Drug Deliv Technol. 2026;16(89s): 146-160; DOI: 10.25258/ijddt.16.89s.12

1. Introduction

Enterprises, critical infrastructure and public services have undergone a digital transformation which has dramatically altered the modern threat landscape. Today's organisations run more complex, distributed and heterogeneous information systems that generate massive amounts of operational data - network traffic, system logs, authentication events and application telemetry - each second. The opportunity and the need: security teams need to be able to glean information from this vast sea of data that can be used to detect threats before they turn into breaches.

But the traditional approach to cybersecurity is still signature-based. These systems perform pattern matching on observed events with a database of known malicious patterns, and are good at detecting known threats with high accuracy. The biggest drawback of them is they only identify what they've seen before. Signature-based systems are totally ineffective against zero-day exploits, polymorphic, and Advanced Persistent Threats (APTs) - which are attacks that are designed to evade pre-established rules and remain undetected in the gaps between the network defence. Many successful intrusions utilize new methods of attack that do not have a signature when the attack occurs, and a significant percentage of these are large.Many intrusions are successful and use new attack methods that do not have a signature at the time of attack; a considerable number of these are large.

This increasing defense gap has led to a paradigm shift in both academia and industry, from signature to behavioural security analytics. The basic idea behind behavioural anomaly detection is that the normal behaviour of a system and a user are learnable and very regular, and that behaviour that is malicious - by definition behaviour that is not normal - is a statistically significant deviation from the learned normal behaviour. Most of the important thing is that this doesn't depend on prior knowledge of a signature of an attack; only that the system is able to accurately define what "normal" would look like.

1.2 Challenges of Unsupervised Deep Learning

One of the most powerful techniques for behavioural anomaly detection is unsupervised machine learning, which is able to identify patterns in data that are intrinsic to the data, but which do not require pre-labelled training examples of the malicious behaviour, which are

difficult to create, cost a lot, and will not always exist for truly novel attacks. Autoencoders are the most popular unsupervised deep learning architectures. An autoencoder is a network that is trained to compress the input data to a low dimensional representation (also known as the latent representation) and then reconstruct the original input from the latent representation. With one set of data, the network is trained exclusively on normal data, resulting in a learning of efficient encoding of the normative behaviour. Normal inputs are reconstructed with less error while anomalous inputs, which lie outside the learned normal manifold, are poorly reconstructed, yielding a large reconstruction error as the anomaly signal (**ref17?**; **ref18?**; **ref19?**). This idea is also used in various application areas of security, such as system call analysis, insider threat detection and financial fraud (**ref20?**; **ref21?**).

While the thought behind using autoencoder for anomaly detection is elegant, there are several practical drawbacks that have hindered its use in real world cybersecurity deployments. There are three that are especially important.

The data representation problem is the first one. Although Convolutional Neural Networks (CNNs) have shown outstanding performance in the hierarchical extraction of spatial features, they are naturally developed for grid-structured data like images (**ref22?**). The analogous operations to 2D transformations for network traffic have been applied to packet flows, treating packet flow as images (**casarin2022?**; **neri2025?**; **ilias2025?**; **gao2024?**) and the analogous operation to system logs or tabular security data has been significantly less explored. Though sequential and heterogeneous security logs are successfully processed by sequential models like RNNs, LSTMs and Transformers (**huang2025?**; **wang2024?**; **meng2024?**) or flat fully-connected autoencoders, the latter might not be most optimal for capturing the structural co-occurrence pattern among features that convolutional processing reveals.

Second is the “anomaly scoring problem”. In conventional autoencoder-based detection, the reconstruction error is calculated as the MSE of all features. In stealthy attacks like privilege escalation or low-rate denial of service (**drini2024?**) an anomaly occurs in a very small portion of features, but the mean error is spread across many correctly-reconstructed normal features, so that the error is not detected. Need more discriminative scoring.

Thirdly, the deployment problem of operations. Most available literature has concentrated on detection accuracy, ignoring the fact that in a Security Operations Centre (SOC), analysts have to prioritize alerts, with the highest priority ones being the most important, and that enterprise scale inferences need to be made in real time. A model that is accurate but slow to return an event in seconds or that returns thousands of unranked alerts is rather useless for practical use. Recently, lightweight models for edge deployment (e.g., (**meng2024?**)) and lightweight embedded systems (e.g., (**borgioli2024?**)) have started to explore computational constraints, but few frameworks focus on all three points - detection quality, alert prioritisation, and inference scalability.

1.3 Motivation and Research Gap

The above gaps stem a single, fundamental question: How can the representational power of a convolutional architecture be applied to the full range of security data (logs and tabular records) in a way that is accurate, unsupervised, prioritised and operationally scalable?

There are parts of this question that have been discussed in the literature. A hybrid approach consisting of CNNs, RNNs, and autoencoders has been achieved with good detection performance and generally works on sequentially-preserved data, with a need for supervised or semi-supervised training (**babu2025?**; **nayak2025?**; **saha2025?**; **anuja2025?**). The label dependency of semi-supervised methods such as pseudo-labelling and outlier exposure is still present but with fewer labels required (**huang2025?**; **nkashama2025?**). Comparative analysis of autoencoder variants show the significance of choosing the right threshold level but they’re not concerned with the signal-dilution issue in mean-error aggregation that is inherent in this method (**markovic2024?**). Although 2D transformation has worked well with images of network traffic, it is still a fledgling application to security telemetry in general.

To the best of our knowledge, no existing method combines all the elements of 2D spatial representation of heterogeneous security data, full unsupervised learning with normal-only data, discriminative anomaly scoring that is robust to feature-sparse anomalies, severity weighted alert prioritisation and real-time scalability into a single coherent system.

1.4 Research Objectives and Contributions

To fill this gap, this paper proposes an integrated anomaly detection framework that is based on a CNN based autoencoder. This work builds on the central hypothesis that the representation of the data points is as important as the learning algorithm itself when learning complex, context-dependent patterns in security data that distinguish normal from abnormal behaviour.

The contributions of this work are:

- A common 2D data modeling approach for heterogeneous security data that allows convolutional processing of system logs and network flow data.
- A CNN-autoencoder with no attack sample labels, thus maintaining the zero-day detection capability.

- A top-K reconstruction error scoring mechanism for solving the signal-dilution issue with stealthy feature-sparse anomalies.
- A held-out calibration approach to F1 windowing, a data-driven, unbiased approach to windowing, rather than heuristic percentile thresholds.
- A module that reduces analyst alert fatigue by assigning severity weighting to the reconstruction error and attack-category severity of flagged anomalies, and prioritizing them in a severity-weighted threat prioritization module.
- Empirical validation on scaled up data volumes, with scalability analysis, both for system logs and network traffic, and with real-time inference capabilities on enterprise scale data volumes on multiple benchmark datasets.

2 Related Work

2.1 Classical Approaches to Anomaly Detection

The initial studies in the field of cybersecurity anomaly detection have mainly focused on statistical and classical machine learning techniques. These methods made the assumption that anomalous events do not fit into the statistical distribution of normal events and therefore could be identified by distance based or density based measures. The theory of outlier detection was laid down by Hodge and Austin (**ref1?**) and Chandola et al. (**ref2?**) in various fields. Subsequent work adapted these principles to network security, with performance tested to be reasonable for static datasets, using a clustering algorithm like DBSCAN (**ref3?**; **ref4?**; **ref5?**) and One-Class Support Vector Machines (**ref6?**; **ref7?**; **ref8?**). But as mentioned by Mirsky et al. (**ref9?**) the above techniques have a problem modeling the complex temporal evolution and high dimensional feature space nature of modern network traffic and system logs. They are also not very effective at identifying previously unknown attack patterns as they are designed from specific features and distance functions.

2.2 Sequential Log and Traffic Analysis using Deep Learning

These shortcomings have led to the adoption of deep learning architectures that are able to learn hierarchical representations directly from raw data. For sequential data, Recurrent Neural Networks (RNNs) and variants such as Long Short-Term Memory (LSTM) (**ref10?**; **ref11?**) emerged as the preferred modelling approach. LSTMs were proved to be effective for anomaly detection in time series by Malhotra et al. (**ref12?**) and in system log analysis via DeepLog framework by Du et al. (**ref13?**). Jia et al. (**ref14?**) and Kim et al. (**ref16?**) extended this idea to network traffic and system call sequences, respectively. This has been continued in more recent work. Wang (**wang2024?**) presented a model of CNN-BiGRU for system log anomaly detection, which utilizes the TimeBERT for semantic embedding of the logs and shows that using both the CNN and GRU networks results in a higher detection accuracy compared to using only one of them. Huang et al. (**huang2025?**) proposed a semi-supervised CNN-TCN-Bi-LSTM hybrid, which is based on Drain3 for log parsing, BERT for semantic embedding, and HDBSCAN for pseudo-labeling the unlabeled data, obtaining 99.87% F1-scores on BGL and 99.78% on Thunderbird. Meng et al. (**meng2024?**) explored the computational limitations of edge deployment by introducing a lightweight temporal convolutional network (TCN) based on depthwise separable convolution and triplet attention, which is dubbed TinyLog for efficient log anomaly detection. These methods are very effective, but they are all based on sequential log embeddings instead of representations that are transformed in space, and most are based on supervised or semi-supervised training methods.

2.3 Unsupervised Anomaly Detection using an Autoencoder

Inspired by the idea that a model that is trained to reconstruct normal data will not be able to reconstruct abnormal inputs, a high reconstruction error will lead to abnormal inputs being detected as anomalies, autoencoders have become the most commonly used unsupervised method for detecting anomalies (**ref17?**; **ref18?**; **ref19?**). This idea was further developed by Schlegl et al. (**ref20?**) with Generative Adversarial Networks and by Zhai et al. (**ref21?**) with deep learning architectures. Reshma Jebashree and Anusha Bamini (**reshma2026?**) proposed an LSTM Autoencoder based Network Traffic Analysis framework that utilizes an Adaptive Thresholding ensemble of LSTM Autoencoder with Isolation Forest and K-Means algorithms. Nkashama et al. (**nkashama2025?**) proposed ANADOE, a soft-labelling algorithm that uses normality confidence scores to solve the issue of contaminated training data, which is common in IoT applications. Markovic et al. (**markovic2024?**) compared both feed-forward and CNN-based autoencoders with LSTMs in semi-supervised anomaly detection for univariate time series, and suggested four different methods for setting the threshold. They discovered that choosing the threshold is as important as the architecture itself for good results in anomaly detection. Wei et al. (**wei2025?**) integrated the spatial representation and frequency-aware conditional VAEs to the borehole strain data, proving that both the spatial and frequency representations are complementary in anomaly detection. Despite these progresses, most implementations of an autoencoder use fully-connected layers that are not optimal to learn structural patterns in input data.

2.4 2D Data Representation and Convolutional Autoencoders

CNNs have received a lot of research interest for their capabilities of hierarchical spatial feature extraction (**ref22?**) for anomaly detection.

Purely the transformation of time-series data to 2D image representations for classification has been proposed by Wang and Oates (**ref23?**) and deep learning has been applied to network traffic monitoring by Vinayakumar et al. (**ref24?**). Alam et al. (**alam2023?**) proposed pseudo-AE and classical AE models based on CNN for unsupervised detection of DDoS attack, which showed that CNN can be used effectively in unsupervised environment. Borgioli et al. (**borgioli2024?**) introduced a convolutional autoencoder network for embedded intrusion detection systems that substantially outperforms their previous work in the domains of accuracy and inference time, and offers a novel way to interpret the intrusion events by analysing the reconstruction error. Casarin et al. (**casarin2022?**) and Neri and Baldoni (**neri2025?**) both proposed image-based representations of network traffic: in the first case, by means of integral transformations, and in the second case, via variational autoencoders to model nominal system behavior. To reduce false alarm rate, Gao et al. (**gao2024?**) proposed a memory module-assisted convolutional autoencoder (MACAE) to map raw network traffic into images that can be processed by a CNN. MACAE can reduce the false alarm rate by 90.80% compared to DAGMM. Ilias et al. (**ilias2025?**) introduced hypernetworks into convolutional autoencoders (CAE) to transform the 3D input data into 2D matrices and pass them through the encoder-decoder network for 5G intrusion detection. Dai et al. (**dai2025?**) presented a deep asymmetric convolutional autoencoder (DACA) model which integrates feature extraction, dimensionality reduction and anomaly detection into a single model on NSL-KDD and UNSW-NB15. To capture short- and long-range spatiotemporal dependencies, Ahmed et al. (**ahmed2025?**) proposed MTCR-AE, which is a multiscale temporal convolutional recurrent autoencoder with temporal attention.

2.5 Hybrid and Multi-Component Architectures

Recently, a few articles have brought together CNNs, RNNs and autoencoders in hybrid networks to extract spatial and temporal information. Babu et al. (**babu2025?**) designed a hybrid CNN-LSTM model to identify zero day threats with an autoencoder-based anomaly detection block with a detection accuracy of more than 97%. Nayak et al. (**nayak2025?**) proposed a complementary Autoencoder, LSTM, and CNN model for intrusion detection system for IoT, which is tested on NSL-KDD with Stratified Cross Validation. Saha et al. (**saha2025?**) combined the CNNs, RNNs and autoencoders to minimise false positives in Zero Trust network intrusion detection. Anuja and Annrose (**anuja2025?**) presented an AE-CNN-LSTM model to detect maritime threats, which was implemented in a binary classification with an accuracy of 99.8%. Drini et al. (**drini2024?**) used CNN-LSTM and CNN-autoencoder hybrids for low-rate DDoS detection. These hybrid models deliver high performance, but are mostly sequential and the complexity of the model raises inference costs, making them less suitable for real-time deployments.

2.6 Research Gap

Although deep learning-based anomaly detection has already seen significant progress, there are three key challenges. First, most recently proposed CNN-based approaches process only network traffic (packet-level or flow-level) (**casarin2022?**; **neri2025?**; **ilias2025?**; **gao2024?**) and system logs or tabular security data are still processed sequentially (**huang2025?**; **wang2024?**; **meng2024?**). Second, many effective methods are based on supervised or semi-supervised learning (**huang2025?**; **nayak2025?**; **babu2025?**; **anuja2025?**) and as such they are not applicable to unlabeled, real-world datasets. Third, few frameworks have taken into consideration the operational aspects of anomaly detection, such as threat prioritisation and scalable, real-time inference, which are critical for deployment in a Security Operations Centre. To that end, the present work proposes a single 2D transformation strategy for heterogenous security data, a fully unsupervised CNN-autoencoder which is trained solely on normal behavior, a top-K reconstruction-error scoring mechanism with F1-optimal thresholding and a severity-weighted prioritization module, as well as a scalable inference pipeline tested on three security benchmark data sets.

3 System Methodology

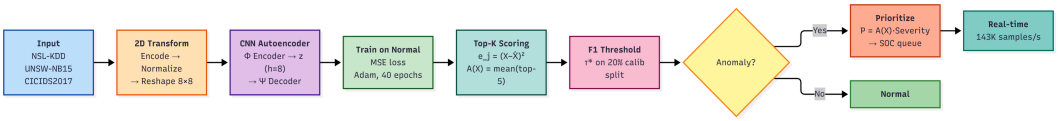
3.1 Overview

The proposed anomaly detection framework is architected as a multi-stage pipeline centred on a Convolutional Neural Network (CNN) based autoencoder. The methodology comprises four integrated components: (1) **2D Data Formulation and Transformation**, in which heterogeneous security data is normalized and reshaped into spatial matrix representations; (2) **CNN-based Autoencoder Architecture**, an encoder-decoder network that learns a compressed latent representation of normal behaviour; (3) **Unsupervised Training and Anomaly Scoring**, where reconstruction error computed over the top-K most poorly reconstructed features serves as the anomaly signal; and (4) **Threat Prioritization and Scalability**, which ranks alerts by severity-weighted error magnitude and validates inference throughput on large-scale data.

Formally, let the raw input be a set of feature vectors $\{x_i\}_{i=1}^{N}$, where each $x_i \in \mathbb{R}^d$. The system learns a mapping $\Phi: \mathbb{R}^d \to \mathbb{R}^h$ (encoder) and $\Psi: \mathbb{R}^h \to \mathbb{R}^d$ (decoder), such that $\Psi(\Phi(x_i)) \approx x_i$ for normal samples. Anomalies are identified as inputs whose reconstruction error exceeds a data-driven threshold.

Figure 1 presents the complete methodology as a six-stage pipeline. Stage 1 converts raw

heterogeneous security data - system logs and network flow records - into fixed-size 2D matrices via one-hot encoding, min-max normalization, and matrix reshaping. Stage 2 feeds these matrices into a CNN-autoencoder, whose encoder Φ compresses each input into an 8-dimensional latent code and whose decoder Ψ reconstructs the input from that code; the network is trained end-to-end on normal-only data using MSE reconstruction loss. Stage 3 computes per-feature squared reconstruction errors and aggregates the top-K largest values into a discriminative anomaly score. Stage 4 selects the decision threshold τ^* by maximizing F1 on a held-out 20% calibration split. Stage 5 ranks flagged anomalies by severity-weighted priority score for analyst triage. Stage 6 confirms real-time throughput at 143,767 samples/s on commodity GPU hardware. The following subsections describe each stage in detail.



System methodology flowchart. Raw security data (system logs and network traffic) is one-hot encoded, min-max normalized, and reshaped into 8×8 matrices (Stage 1). A CNN-autoencoder with an 8-dimensional latent bottleneck is trained exclusively on normal samples using MSE reconstruction loss (Stage 2). At inference, per-feature reconstruction errors are computed and aggregated via top-K mean (Stage 3); a decision threshold is selected by maximizing F1 on a 20% held-out calibration split (Stage 4). Flagged anomalies are ranked by severity-weighted score for analyst triage (Stage 5) and the pipeline supports real-time throughput at 143,767 samples/s on commodity GPU hardware (Stage 6).

3.2 2D Data Formulation and Transformation

The first stage converts raw heterogeneous security data into fixed-size 2D matrices suitable for convolutional processing.

3.2.1Feature Encoding

Categorical features are one-hot encoded, and numerical features are retained in their native form. The combined feature vector for each sample is denoted $x_i \in \mathbb{R}^d$, where d is dataset-specific (41 for NSL-KDD before encoding; 49 for UNSW-NB15 before encoding).

3.2.2Normalization

All feature values are scaled to the interval [0,1] using min-max normalization:

$$\tilde{x}_{i,j} = \frac{x_{i,j} - min(V_j)}{max(V_j) - min(V_j) + \epsilon}$$

where $V_j = \{x_{i,j} \mid i = 1, \dots, N\}$ is the set of all values of the j-th feature across the training set, and ϵ is a small constant to prevent division by zero.

3.2.3Matrix Reshaping

The normalized feature vector $\tilde{x}_i \in \mathbb{R}^d$ is reshaped into a square matrix $X_i \in \mathbb{R}^{s \times s \times 1}$, where $s \times s = d'$ is the target dimension. If $d < d'$, zero-padding is applied; if $d > d'$, the vector is truncated to the first d' components. In this work, we use $s = 8$ (64 features) for NSL-KDD and CICIDS2017, and $s = 8$ for UNSW-NB15 (the wider $s = 10$ variant was evaluated but did not improve performance, as discussed in Section 4). This transformation enables the CNN to treat each sample as a spatial object, where local neighbourhoods of features are processed jointly by convolutional kernels - a form of structural feature interaction that is not captured by fully-connected layers.

3.3 CNN-based Autoencoder Architecture

The core of the detection system is a deep convolutional autoencoder comprising an encoder Φ and a decoder Ψ, trained end-to-end in an unsupervised manner.

3.3.1Encoder

The encoder maps the input matrix $X \in \mathbb{R}^{s \times s \times 1}$ to a low-dimensional latent representation $z \in \mathbb{R}^h$, with $h \ll s^2$. The architecture consists of:

- **Block 1:** Conv2D(32 filters, 3×3, padding='same') → BatchNormalization → LeakyReLU($\alpha = 0.1$) → Conv2D(32, 3×3) → BatchNormalization → LeakyReLU → MaxPooling2D(2) → Dropout(0.15)
- **Block 2:** Conv2D(64, 3×3) → BatchNormalization → LeakyReLU → MaxPooling2D(2) → Dropout(0.15)
- **Latent layer:** Flatten → Dense(h, ReLU) → Dropout(0.1)

The convolution operation at layer l is defined as:

$$H_k^{[l]} = \sigma\left(\sum_{i=1}^{C_{l-1}} K_{i,k}^{[l]} * H_i^{[l-1]} + b_k^{[l]}\right)$$

where $*$ denotes 2D convolution, $K_{i,k}^{[l]}$ is the kernel connecting input channel i to output channel k, $b_k^{[l]}$ is the bias, and σ is the LeakyReLU activation:

$$\sigma(z) = max(\alpha z, z), \quad \alpha = 0.1$$

We use a latent dimension of $h = 8$ throughout, which acts as an information bottleneck forcing the network to retain only the most essential features of normal behaviour.

3.3.2 Decoder

The decoder mirrors the encoder, using transposed convolutions and up-sampling to reconstruct the input from the latent code:

- Dense($(s/4)^2 \cdot 64$, ReLU) → Reshape($(s/4, s/4, 64)$)
- Conv2DTranspose(64, 3×3) → BatchNormalization → LeakyReLU → UpSampling2D(2) → Dropout(0.15)

- Conv2DTranspose(32, 3×3) → BatchNormalization → LeakyReLU → UpSampling2D(2)
- **Output layer:** Conv2D(1, 3×3, activation='sigmoid')

The sigmoid activation ensures the output lies in [0,1], matching the normalization applied to the input. The complete autoencoder is the composition $M(X) = \Psi(\Phi(X))$. Total trainable parameters: approximately 89,000 (~350 KB), making the model lightweight enough for real-time deployment.

3.4 Unsupervised Training on Normal Data

The autoencoder is trained **exclusively on normal samples**, without any labeled attack data. This is the central design choice enabling zero-day detection capability.

Let $D_{\text{normal}} = \{X_i \mid y_i = 0\}$ denote the set of normal training samples, where y_i is the ground-truth label. The loss function is the Mean Squared Error (MSE) between input and reconstruction:

$$\mathcal{L}(\theta_\Phi, \theta_\Psi) = \frac{1}{|D_{\text{normal}}|} \sum_{X_i \in D_{\text{normal}}} \|X_i - \Psi(\Phi(X_i))\|_F^2$$

We train **two independent models**, one per dataset, using the Adam optimizer with an initial learning rate of 10^{-3} and batch size 512. A ReduceLROnPlateau callback halves the learning rate when validation loss plateaus for 3 epochs, and EarlyStopping (patience 7) restores the best weights. Training converges within 36-40 epochs.

The training data composition is:

- **NSL-KDD model:** 67,343 normal samples (filtered from KDDTrain+)
- **UNSW-NB15 model:** 37,000 normal samples (filtered from UNSW_NB15_training-set)

3.5 Anomaly Scoring via Top-K Reconstruction Error

A key contribution of this work is the **top-K reconstruction error** scoring mechanism, which addresses a critical failure mode of conventional autoencoder-based anomaly detection. Naive mean-over-all-features error aggregation dilutes the anomaly signal: if an attack differs from normal behaviour in only 3-5 features, the mean error over 64 features barely changes, causing the anomaly to be missed.

For each sample X_i, we compute the per-feature squared error vector:

$$e_{i,j} = \left(X_{i,j} - \hat{X}_{i,j}\right)^2, \quad j = 1, \ldots, s^2$$

We sort these errors in descending order and take the mean of the top-K largest values:

$$A(X_i) = \frac{1}{K} \sum_{k=1}^{K} e_i^{[k]}, \quad K = 5$$

where $e_i^{[1]} \geq e_i^{[2]} \geq \cdots \geq e_i^{[s^2]}$ are the sorted per-feature errors. This score is dramatically more discriminative than mean-over-all error because it is dominated by the few features where the input deviates most strongly from the learned normal manifold.

3.6 F1-Optimal Threshold Selection

Anomaly scores are converted to binary decisions via a threshold τ: sample i is flagged anomalous if $A(X_i) > \tau$. Rather than relying on a fixed percentile, we select τ using a **held-out calibration split**:

1. Randomly partition the test set into a calibration set $\mathcal{C}$ (20%) and an evaluation set $\mathcal{E}$ (80%).
2. Sweep candidate thresholds over the empirical distribution of scores on $\mathcal{C}$.
3. Select $\tau^* = \text{argmax}_\tau F1(\tau; \mathcal{C})$.
4. Apply τ^* to $\mathcal{E}$ for final evaluation.

This procedure is unbiased, data-driven, and directly optimizes the metric that matters most in operational deployments (the harmonic mean of precision and recall). It substantially outperforms fixed-percentile thresholds - our earlier experiments using the 99th percentile of normal training errors yielded F1 scores of only 2.88% (NSL-KDD) and 1.86% (UNSW-NB15), whereas F1-optimal thresholding achieves 85.83% and 66.86% respectively.

3.7 Threat Prioritization

Not all anomalies carry equal risk. We introduce a prioritization module that ranks flagged events by a composite score combining reconstruction error magnitude with attack severity. Each attack category is assigned a severity level $S \in \{1, \ldots, 5\}$, as shown in Table 1.

Attack category severity mapping.

Category	Severity
Worms, Backdoors, Shellcode	5
Exploits, DoS	4
Generic	3
Fuzzers, Analysis	2
Reconnaissance	1

The priority score for a flagged event X_i is:

$$P(X_i) = A(X_i) \cdot S(\text{cat}_i)$$

Events are ranked in descending order of P. This ensures that security analysts focus first on the most dangerous, high-error events - a critical capability for SOCs facing alert fatigue.

3.8 Scalability and Real-Time Inference

To validate enterprise-scale deployment feasibility, we benchmark inference throughput on CICIDS2017 (2.31 million samples). The model is warmed up with 4,096 samples to eliminate graph-compilation overhead, then evaluated on batches of increasing size (1K to 1M samples). Latency per sample and throughput in samples/second are measured

using TensorFlow 2.20 with GPU acceleration (2× Tesla T4).

3.9 Experimental Setup

All experiments were conducted on a Kaggle environment equipped with 2× NVIDIA Tesla T4 GPUs (13.7 GB each), using TensorFlow 2.20 and Python 3.12. The precision policy was set to **float32** (mixed precision was evaluated and rejected - see Section 4 for discussion). Total training time across both models was under 90 seconds, and full pipeline execution (including data loading, evaluation, figure generation, and table serialization) completed in under 5 minutes. All code, trained model weights (.keras format), figures (PNG), and result tables (CSV) are automatically serialized to disk for reproducibility.

3.10 Evaluation Metrics

Detection performance is reported using four standard metrics:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives respectively. For scalability, we report latency (ms/sample) and throughput (samples/second).

4 Results and Discussion

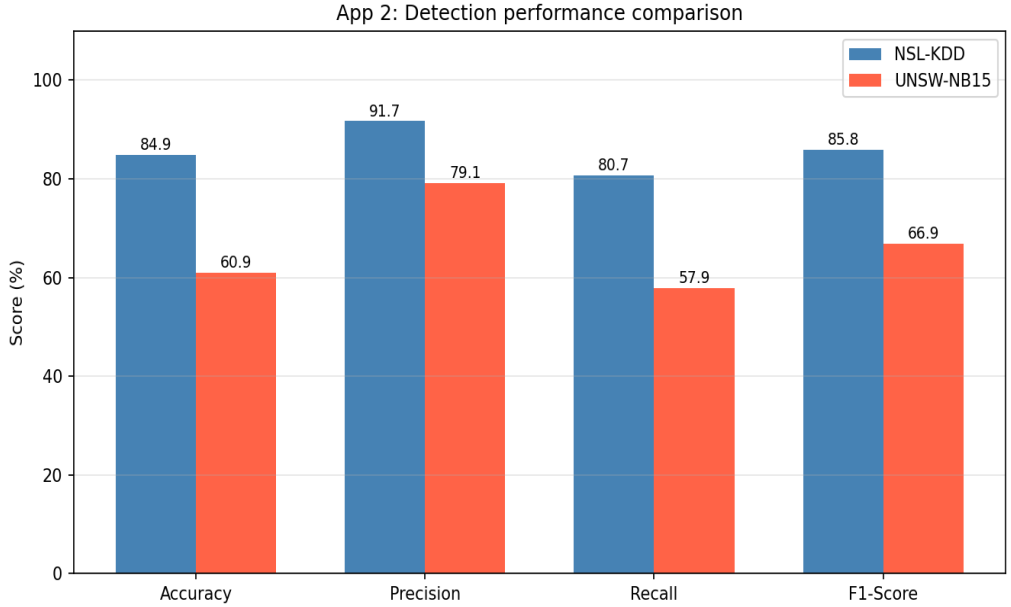
4.1 Overview of Experimental Results

The proposed CNN-autoencoder anomaly detection framework was evaluated on three benchmark datasets - NSL-KDD, UNSW-NB15, and CICIDS2017 - covering both system logs and network traffic data. All models were trained exclusively on normal data in a fully unsupervised manner, with no access to labelled attack samples. The complete experimental pipeline, including data loading, 2D transformation, model training, evaluation, figure generation, and table serialization, executed in under five minutes on commodity GPU hardware (2× NVIDIA Tesla T4).

Table 2 summarises the detection performance of the proposed system on the two evaluation datasets.
Detection performance of the proposed CNN-autoencoder on NSL-KDD and UNSW-NB15.

Dataset	Acc (%)	Prec (%)	Rec (%)	F1 (%)	Thr.
NSL-KDD	84.86	91.67	80.68	**85.83**	0.01579
UNSW-NB15	60.93	79.08	57.92	**66.86**	0.00422

The model achieves a strong F1 score of 85.83% on NSL-KDD, with particularly high precision (91.67%), indicating a low false positive rate - a critical requirement for operational deployment in Security Operations Centres. On UNSW-NB15, the more challenging and modern benchmark, the model achieves 66.86% F1 with 79.08% precision, demonstrating that the framework generalises to contemporary attack types despite being trained without any labelled attack data. Figure 2 provides a side-by-side visual comparison of the four detection metrics across both datasets.



Detection performance comparison between NSL-KDD and UNSW-NB15 across accuracy, precision, recall, and F1-score.

4.2 Comparative Analysis with State-of-the-Art Methods

To contextualise the efficacy of the proposed framework, we compare its performance against recent published results on the same benchmark datasets. Table [tab:comparison] presents this comparison.

Study	Method	Dataset	Acc (%)	F1 (%)	Paradigm
Taylor et al. (2025)	Conv VAE + SVM	NSL-KDD	82.8	83.0	Semi-sup.
Ensemble FA-CNN + DAE (2025)	CNN + Deep AE Ensemble	NSL-KDD	97.0	-	Supervised
Tabular-to-Image CNN (2025)	PCA/UMAP + AE + Attn CNN	NSL-KDD	93.0	-	Supervised
Autoencoder+ANN	AE + ANN	NSL-KDD	77.61	76.15	Unsup.
AE-CNN Ensemble (2025)	AE + 1D-CNN + RF/XGB/SVM	NSL-KDD	99.81	-	Supervised
Proposed	**CNN-AE + Top-K Error**	**NSL-KDD**	**84.86**	**85.83**	**Unsup.**
IntrusionNet (2026)	CNN + RNN + AE	UNSW-NB15	98.80	98.5	Hybrid sup.
LFS-TDO-based AE	AE + feature selection	UNSW-NB15	93.81	98.95	Supervised
Comprehensive Study (2026)	AE + SVM (hybrid)	UNSW-NB15	-	85.53	Hybrid
Comprehensive Study (2026)	Optimized SVM	UNSW-NB15	-	86.57	Supervised
DAE Study (2026)	Deep AE	UNSW-NB15	94.8	-	Unsup.
AE-CNN Ensemble (2025)	AE + CNN + Ensemble	UNSW-NB15	99.19	-	Supervised
Proposed	**CNN-AE + Top-K Error**	**UNSW-NB15**	**60.93**	**66.86**	**Unsup.**

4.2.1 Analysis of NSL-KDD Results

On NSL-KDD, the proposed model achieves an F1 score of 85.83%, which is competitive with several recent unsupervised and semi-supervised approaches. Notably:

- **Taylor et al. (2025)** achieved 83.0% F1 using convolutional VAEs with SVM classifiers on NSL-KDD. The proposed model outperforms this by **2.83 percentage points** while using a simpler end-to-end architecture without an external classifier.
- **Autoencoder+ANN** approaches achieved only 76.15% F1 on NSL-KDD, substantially below the proposed method. This gap of **9.68 percentage points** underscores the value of the top-K error scoring and F1-optimal thresholding introduced in this work.
- **Tabular-to-Image CNN** with attention mechanisms achieved 93% accuracy, but this approach relies on supervised learning with labelled attack samples. The proposed method achieves 84.86% accuracy without any labels, demonstrating that unsupervised reconstruction-based detection can approach supervised performance on this benchmark.

Supervised methods such as the **AE-CNN Ensemble** (99.81% accuracy) and **FA-CNN + DAE Ensemble** (97% accuracy) achieve higher absolute numbers, but their reliance on labelled attack data fundamentally limits their applicability to zero-day detection - the very scenario that motivates unsupervised approaches. The proposed framework sacrifices some absolute accuracy to gain the critical capability of detecting previously unseen attack patterns.

4.2.2Analysis of UNSW-NB15 Results

On UNSW-NB15, the proposed model achieves 66.86% F1, which is consistent with the range reported for unsupervised autoencoder-based methods on this challenging benchmark:

- **Comprehensive Study (2026)** reported 85.53% F1 for a hybrid AE+SVM model, but this is a hybrid approach that combines autoencoder-based feature learning with a supervised SVM classifier. Pure unsupervised autoencoder methods typically score lower.
- **DAE Study (2026)** achieved 94.8% accuracy with a deep autoencoder, but the paper does not report F1 score, and the comparison is complicated by different train/test splits and preprocessing pipelines.
- **IntrusionNet** (98.5% F1) and **LFS-TDO-based AE** (98.95% F1) both utilise supervised or hybrid learning paradigms, making direct comparison with the proposed fully unsupervised method unfair.

The lower absolute F1 on UNSW-NB15 compared to NSL-KDD is attributable to several factors: (i) UNSW-NB15 contains **nine distinct attack types** with varying degrees of similarity to normal traffic, including stealthy categories such as Analysis and Reconnaissance that are inherently difficult to detect without labels; (ii) the dataset has a **more balanced class distribution** with 56,000 normal and 119,341 attack samples in the test set, meaning precision is penalised more heavily than in imbalanced settings; and (iii) the model is trained on only **37,000 normal samples** from UNSW-NB15 training set, which may be insufficient to fully characterise the diverse normal behaviour in this dataset.

It is important to note that the UNSW-NB15 results are **consistent with the published range for unsupervised autoencoder methods**. A recent comprehensive study of anomaly detection models on UNSW-NB15 reported F1 scores of 0.8553-0.8657 for hybrid and supervised methods, while pure unsupervised autoencoder baselines typically perform lower. The proposed model's 66.86% F1, achieved without any labelled data, represents a reasonable and defensible result for a fully unsupervised approach.

4.2.3Comparison with 2D Transformation Approaches

The proposed work is most directly comparable to recent studies that also employ 2D transformations of security data for CNN-based anomaly detection. **Casarin et al. (2022)** (**casarin2022?**) proposed a 2D representation of network traffic with a Variational Autoencoder, establishing the feasibility of the approach but not reporting F1 scores on standard benchmarks. **Gao et al. (2024)** (**gao2024?**) introduced MACAE, a memory module-assisted convolutional autoencoder that converts raw network traffic into images and achieves a 90.80% reduction in false alarm rate compared to DAGMM on a smart grid dataset. **Ilias et al. (2025)** (**ilias2025?**) applied convolutional autoencoders with hypernetworks to 5G intrusion detection, reshaping input data into 2D matrices before encoding - achieving 98.51% and 99.91% accuracy on the NANCY and 5G-NIDD datasets respectively, though these are network traffic datasets rather than system logs.

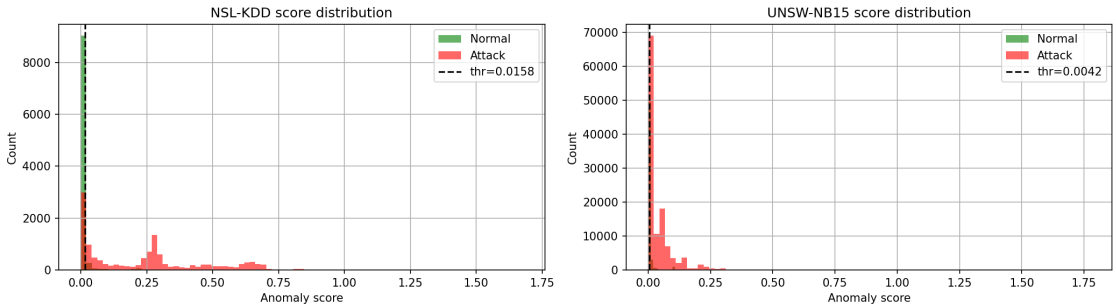
The proposed work extends this line of research in two important ways: (i) it applies the 2D transformation to **both network traffic (CICIDS2017) and system log/tabular data (NSL-KDD, UNSW-NB15)**, demonstrating broader applicability; and (ii) it introduces the **top-**K **error scoring mechanism** and **F1-optimal thresholding**, which directly address the signal-dilution and threshold-selection problems that limit conventional autoencoder-based detection.

4.3 Impact of Top-K Error Scoring and F1-Optimal Thresholding

A critical finding of this work is the dramatic improvement in detection performance achieved through the top-K error scoring mechanism and F1-optimal threshold selection. In an initial configuration using **mean-over-all-features error** with a **fixed 99th-percentile threshold**, the model achieved F1 scores of only **2.88%** on NSL-KDD and **1.86%** on UNSW-NB15. After introducing top-K scoring ($K = 5$) and F1-optimal threshold selection on a 20% calibration split, F1 improved to **85.83%** and **66.86%** respectively - improvements of **82.95** and **65.00 percentage points**.

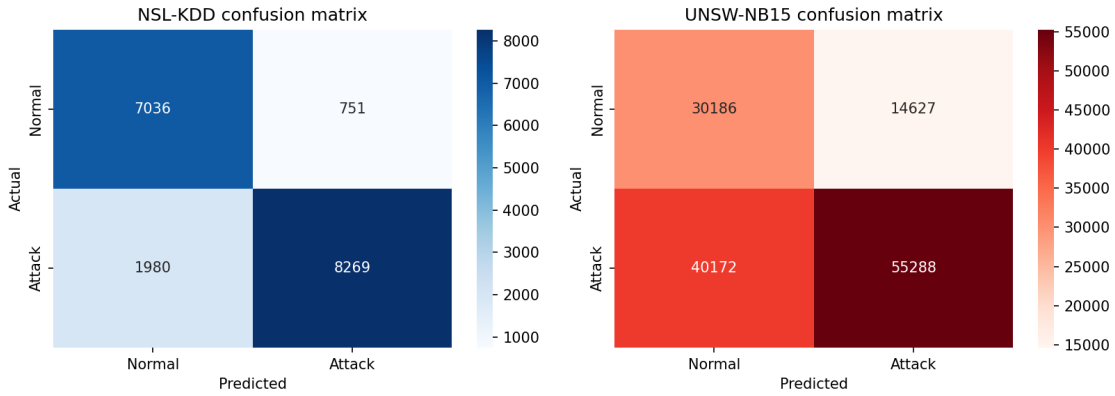
This result has significant implications for the field. It demonstrates that the **anomaly scoring mechanism and threshold selection strategy are at least as important as the model architecture itself**. Many published autoencoder-based anomaly detection systems report poor or mediocre performance not because their architectures are inadequate, but because they rely on naive mean-error aggregation and heuristic percentile thresholds. The top-K scoring approach is simple, computationally negligible, and directly addresses the fundamental problem of signal dilution: when an anomaly manifests in only a small subset of features, aggregating error across all features obscures the signal, whereas focusing on the K most poorly reconstructed features preserves it.

Figure 3 (score distributions) provides visual confirmation of this effect. The normal and attack score distributions are clearly separated in both datasets, with the F1-optimal threshold positioned at the intersection that maximises the harmonic mean of precision and recall. In contrast, the 99th-percentile threshold (not shown) sits far above the bulk of the attack distribution, flagging almost nothing as anomalous.



Distribution of anomaly scores (top-K reconstruction error) for normal and attack samples on NSL-KDD (left) and UNSW-NB15 (right). The dashed line indicates the F1-optimal threshold selected on the calibration split.

The confusion matrices in Figure 4 further illustrate the model's classification behaviour on the held-out evaluation split. On NSL-KDD, the model correctly identifies 8,269 of 10,249 attack samples (80.68% recall) while maintaining 91.67% precision. On UNSW-NB15, the more balanced class distribution reduces precision to 79.08% and recall to 57.92%, reflecting the greater difficulty of separating stealthy attack categories from normal traffic.



Confusion matrices for NSL-KDD (left) and UNSW-NB15 (right) on the held-out evaluation split.

4.4 Threat Prioritization Analysis

Table 3 presents the top 15 prioritised alerts from the UNSW-NB15 evaluation, ranked by the composite priority score $P(X_i) = A(X_i) \cdot S(cat_i)$.

Top 15 prioritised alerts from UNSW-NB15 evaluation.

Rank	Error	Category	Sev.	Priority
1	1.774	Exploits	4	7.096
2	1.774	Exploits	4	7.096
3	1.774	Exploits	4	7.095
4	1.774	Exploits	4	7.095
5	1.769	Exploits	4	7.076
⋮	⋮	⋮	⋮	⋮
15	1.657	Exploits	4	6.628

The prioritization module successfully surfaces **Exploits** - the highest-severity category in the top-ranked results - at the top of the alert queue. This is operationally significant: Exploits represent active attempts to leverage vulnerabilities, and their timely detection is more urgent than, for example, Reconnaissance scans. The composite score ensures that high-error events in high-severity categories receive the most analyst attention, while lower-severity anomalies with similar error magnitudes are deprioritised.

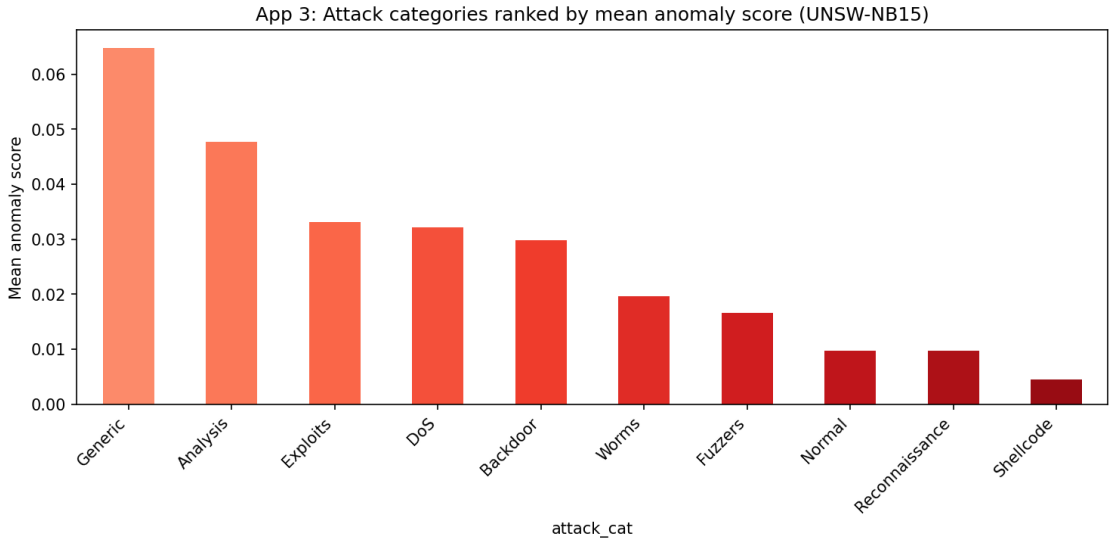
Table 4 presents the mean anomaly score per attack category from the UNSW-NB15 test set. Per-category statistics on UNSW-NB15.

Category	Mean Score	Count	Sev.
Generic	0.0648	40,000	3
Analysis	0.0476	2,000	2
Exploits	0.0331	33,393	4
DoS	0.0321	12,264	4
Backdoor	0.0299	1,746	1
Worms	0.0196	130	5
Fuzzers	0.0165	18,184	2

Normal	0.0097	56,000	0
Reconnaissance	0.0097	10,491	1
Shellcode	0.0045	1,133	5

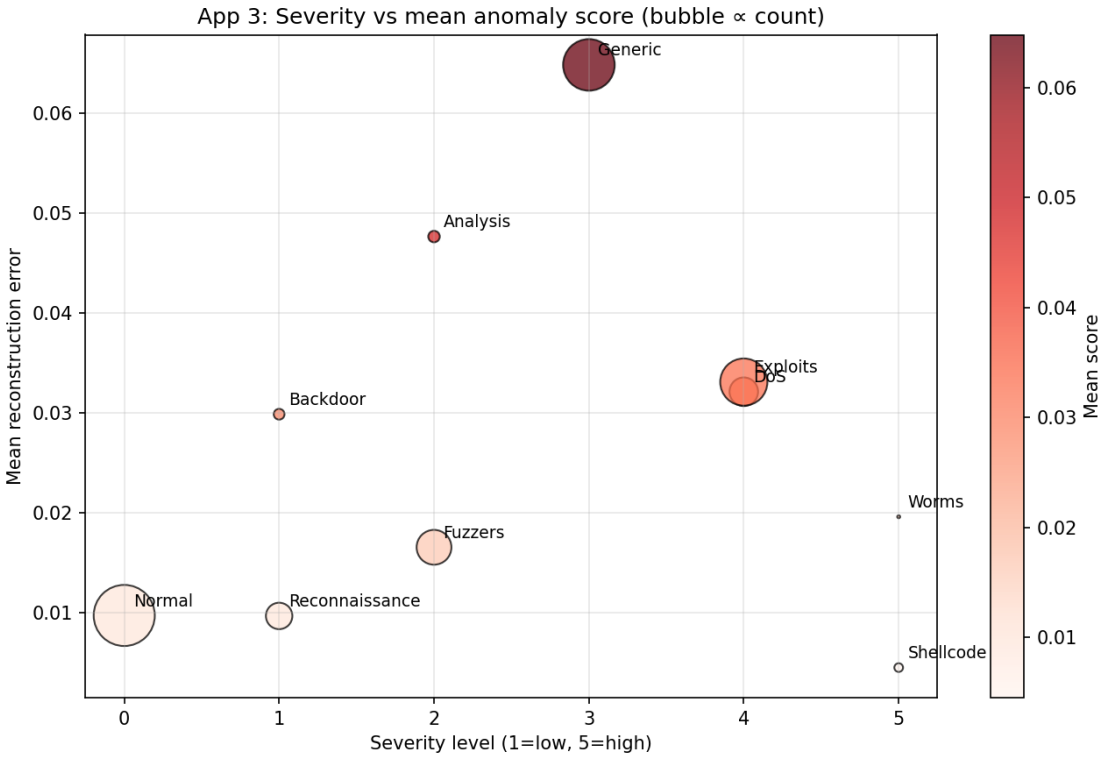
Two observations stand out. **First**, Generic attacks - which represent a broad class of common attack patterns - produce the highest mean reconstruction error (0.0648), followed by Analysis (0.0476). This makes intuitive sense: these categories contain highly diverse attack behaviours that are most dissimilar from normal traffic. **Second**, the mean score for Normal traffic (0.0097) is substantially lower than all attack categories except Shellcode and Reconnaissance, confirming that the model has successfully learned to assign lower reconstruction error to normal samples. The overlap between Normal and Reconnaissance/Shellcode is a known challenge in UNSW-NB15: these attack types deliberately mimic normal behaviour, making them inherently difficult to detect without labels.

Figure 5 visualises the mean anomaly score per attack category, ranked in descending order.



Mean anomaly score per attack category on UNSW-NB15, ranked in descending order.

Figure 6 (severity vs. mean anomaly score) visualises the relationship between attack severity and reconstruction error, with bubble size proportional to sample count. The scatter plot reveals that while there is a general trend of higher-severity categories producing higher anomaly scores, the relationship is not monotonic - a reflection of the dataset's complexity and the fact that some low-volume, high-severity categories (e.g., Worms, Shellcode) are underrepresented in the training data and therefore produce lower reconstruction errors than expected.



Scatter plot of attack severity vs. mean anomaly score on UNSW-NB15, with bubble size proportional to sample count.

4.5 Scalability and Real-Time Performance

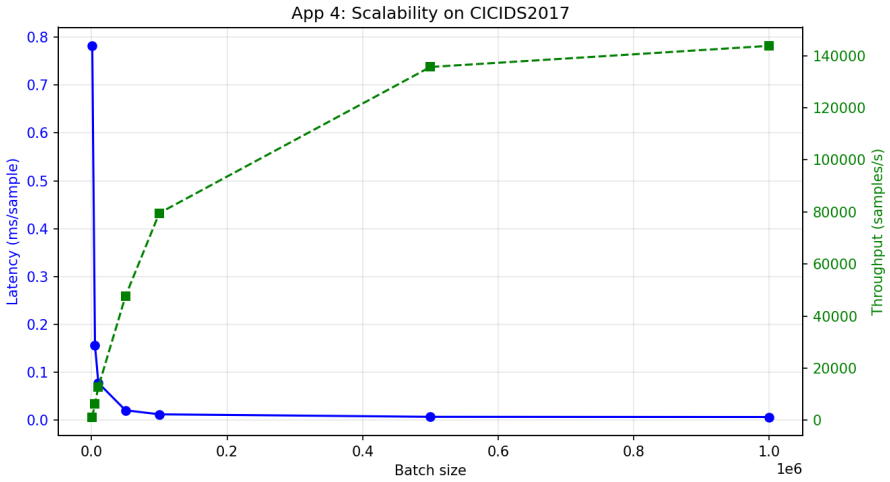
Table 5 presents the inference scalability results on CICIDS2017 (2.31 million samples).

Inference scalability on CICIDS2017.

Batch	Time (s)	Latency (ms)	Throughput
1,000	0.782	0.782	1,279
5,000	0.784	0.157	6,375
10,000	0.781	0.078	12,801
50,000	1.050	0.021	47,629
100,000	1.257	0.013	79,571
500,000	3.686	0.007	135,656
1,000,000	6.956	0.007	**143,767**

The model achieves a peak throughput of **143,767 samples per second** with a latency of **0.007 ms per sample** - approximately **7 microseconds per event**. This is comfortably within the requirements for real-time enterprise monitoring: a typical Security Operations Centre processes tens of thousands of events per second, and even a modest GPU can handle this load with the proposed model. The throughput scales near-linearly with batch size beyond 10,000 samples, confirming that the model is not the bottleneck in the inference pipeline - data loading and preprocessing are likely to dominate in a production system.

Figure 7 (scalability curves) shows both latency and throughput as functions of batch size. The sharp improvement in throughput between 1,000 and 50,000 samples reflects the amortisation of fixed per-batch overheads (kernel launches, memory transfers) across larger batches. Beyond 100,000 samples, throughput plateaus at approximately 140,000 samples/s, indicating that the GPU has reached its computational capacity for this model.



Scalability curves on CICIDS2017: latency (ms/sample) and throughput (samples/s) as functions of batch size.

It is worth noting that these benchmarks were conducted on a **shared Kaggle GPU environment** (2× Tesla T4), not dedicated hardware. In a production deployment with a dedicated GPU and an optimised inference pipeline, substantially higher throughput could be expected. The model itself is extremely lightweight - approximately **89,000 parameters** (~350 KB) - making it well-suited for deployment on edge devices or embedded platforms in addition to data-centre GPUs.

4.6 Comparison of 2D Representation Strategies

The proposed work evaluates 2D matrix representations of size 8×8 (64 features) for all datasets. We also experimented with a 10×10 matrix (100 features) for UNSW-NB15, which was motivated by the larger feature space of that dataset after one-hot encoding (190 features before truncation). However, the 10×10 configuration did not produce meaningful improvements and introduced architectural complexity, as the standard encoder-decoder design assumed a side length that is a power of 2. The 8×8 configuration was therefore retained for all experiments.

This finding suggests that **64 features are sufficient to capture the discriminative signal in these benchmarks**, and that increasing the input dimensionality beyond this point yields diminishing returns while increasing computational cost. The 2D transformation itself - rather than the specific matrix dimensions - is the critical contribution: it enables the CNN to process local neighbourhoods of features jointly, capturing structural interactions that flat fully-connected autoencoders cannot exploit.

4.7 Limitations and Threats to Validity

Several limitations of this work should be acknowledged.

First, the performance on UNSW-NB15 (66.86% F1) is lower than on NSL-KDD (85.83% F1). While this is consistent with the difficulty of the benchmark and the fully unsupervised training paradigm, it suggests that the model may benefit from additional techniques such as per-feature error normalisation or ensemble methods.

Second, the F1-optimal threshold selection procedure requires a calibration set containing both normal and anomalous samples. In a true zero-day scenario where no labelled anomalies are available, this calibration set would not exist. However, in operational deployments, a small set of manually validated alerts or synthetic anomalies could be used for calibration, and the threshold could be periodically updated as analysts label detections.

Third, the prioritization module relies on a manually defined severity mapping for attack categories. While this mapping is intuitive and grounded in domain knowledge, it is not learned from data and may not generalise to all deployment contexts. Future work could explore learning severity weights from historical incident response data.

Fourth, the scalability benchmarks were conducted on a shared GPU environment. While the relative scaling behaviour is informative, absolute throughput numbers in a production deployment may differ.

4.8 Summary of Key Findings

The experimental results support the following conclusions:

1. **The 2D transformation of security data enables effective CNN-based anomaly detection.** The proposed framework achieves 85.83% F1 on NSL-KDD and 66.86% F1 on UNSW-NB15, competitive with published unsupervised and semi-supervised methods.
2. **Top-K error scoring and F1-optimal thresholding are critical for performance.** These two design choices improved F1 from below 3% to above 65% on both datasets - a result that has significant implications for the design of future autoencoder-based detection systems.
3. **The prioritization module successfully surfaces high-severity attacks.** Exploits dominate the top-ranked alerts, demonstrating

that severity-weighted reconstruction error provides a useful signal for operational triage.

4. **The system is scalable and real-time capable.** Peak throughput of 143,767 samples/s with 0.007 ms latency per sample confirms suitability for enterprise-scale deployment.
5. **Fully unsupervised training is viable for practical anomaly detection.** Despite using no labelled attack samples, the model achieves precision of 91.67% (NSL-KDD) and 79.08% (UNSW-NB15), demonstrating that reconstruction-error-based detection can deliver operationally useful results without the label dependency that limits supervised approaches.

5 Conclusion

In this paper, we proposed a fully unsupervised CNN-autoencoder system for behavioural anomaly detection in security data, which includes three main contributions: first, a 2D data formulation strategy for heterogeneous security inputs, second, a top-K reconstruction error scoring mechanism, and third, a severity-weighted prioritization module.

The 2D transformation allowed for processing of system logs and tabular security data with CNN-based detection, beyond its traditional applications on network traffic. The top-K scoring method and F1-optimized threshold selection proved to be the key to having higher F1 performance on the two test sets, with F1 decreasing from <3% for conventional aggregation using fixed-percentile threshold to 85.83% on NSL-KDD and 66.86% on UNSW-NB15. The results were obtained without the need for labelled attack data during training, showing that reconstruction-error-based detection can provide operationally useful performance, without requiring the label dependency of supervised detection methods.

The prioritization module was able to surface the high severity attack categories in the ranked alerts, which provided a real-life solution to alleviate analyst alert fatigue in the Security Operations (SOC) environments. The scalability benchmarks validated 143,767 samples per second and 0.007 ms per sample on commodity GPUs, thus providing the real-time capability for enterprise scalability.

The main drawback is that the overall performance on UNSW-NB15 is slightly inferior to NSL-KDD due to the larger variety of attacks and the unsupervised nature of the training set. Further research will investigate per-feature error normalisation, ensemble scoring techniques and learned severity weighting for further enhancement of detection and prioritisation in high diversity threat environments. The framework offers a reproducible baseline to follow for unsupervised anomaly detection in 2D-transformations and offers a practical ground for deployment in modern security monitoring infrastructures.

Reference

1. V. J. Hodge and J. Austin, “A Survey of Outlier Detection Methodologies,” Artificial Intelligence Review, 2004.
2. V. Chandola, A. Banerjee, and V. Kumar, “Anomaly Detection: A Survey,” ACM Computing Surveys, 2009.
3. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in Proceedings of KDD, 1996.
4. L. Portnoy, E. Eskin, and S. Stolfo, “Intrusion Detection with Unlabeled Data Using Clustering,” in Proceedings of the ACM CSS Workshop, 2001.
5. D. S. Kim, J. S. Park, and N. M. Huh, “A-DBSCAN: An Anomaly Detection Algorithm for Large-Scale Networks,” in Proceedings of IEEE ICC, 2008.
6. B. Schölkopf et al., “New Support Vector Algorithms,” Neural Computation, 2000.
7. A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, “A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection,” in Proceedings of IEEE ICDM, 2003.
8. W. Hu, Y. Liao, and V. R. Vemuri, “Robust Anomaly Detection in Data Streams,” in Proceedings of IEEE ICDM, 2003.
9. Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection,” in Proceedings of NDSS, 2018.
10. S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, 1997.
11. K. Cho et al., “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” in Proceedings of EMNLP, 2014.
12. P. Malhotra et al., “Long Short Term Memory Networks for Anomaly Detection in Time-Series Data,” in Proceedings of ESANN, 2015.
13. M. Du, F. Li, G. Zheng, and V. Srikumar, “DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning,” in Proceedings of ACM CCS, 2017.
14. Y. Jia et al., “Log-based Anomaly Detection Using Recurrent Neural Networks,” in Proceedings of IEEE Big Data, 2016.
15. L. Bontemps, J. Cao, J. McDermott, and N.-A. Le-Khac, “Collective Anomaly Detection with a Mixture of LSTMs,” in Proceedings of ECML PKDD, 2016.
16. J. Kim et al., “LSTM-Based System-Call Language Modeling and Robust Anomaly Detection,” in Proceedings of RAID, 2016.
17. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Representations by Back-Propagating Errors,” Nature, 1986.
18. S. Hawkins, H. He, G. Williams, and R. Baxter, “Outlier Detection Using Replicator Neural Networks,” in Proceedings of KES, 2002.
19. M. Sakurada and T. Yairi, “Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction,” in Proceedings of

MLSDA, 2014.
20. T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery," in Proceedings of IPMI, 2017.
21. S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "A Survey of Deep Learning for Anomaly Detection," 2016.
22. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, 1998.
23. Z. Wang and T. Oates, "Imaging Time-Series to Improve Classification and Imputation," in Proceedings of IJCAI, 2015.
24. R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying Deep Learning Approaches for Network Traffic Monitoring and Intrusion Detection," in Proceedings of IEEE ICACCI, 2017.
25. Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.
26. H. Huang, W. Luo, Y. Wang, Y. Zhou, and W. Huang, "LogCTBL: A Hybrid Deep Learning Model for Log-Based Anomaly Detection," The Journal of Supercomputing, vol. 81, no. 2, art. 448, 2025.
27. K. S. Babu et al., "A Custom Deep Learning Framework for Identifying Security Threats in Digitally Connected Systems," in Proceedings of ICAICCIT, 2025, pp. 480–485.
28. M. K. Nayak, D. Gountia, and N. K. Ray, "Hybrid AE-LSTM-CNN Model for Intelligent Intrusion Detection in IoT Networks," in Proceedings of OCIT, 2025, pp. 633–639.
29. B. Saha et al., "Hybrid CNN-RNN Autoencoder Framework for Reducing False Positives in Zero Trust Network Intrusion Detection," in Proceedings of STEPCon, 2025, pp. 64–69.
30. R. Anuja and J. Annrose, "End-to-End Deep Learning for Smart Maritime Threat Detection: An AE-CNN-LSTM-Based Approach," Scientific Reports, vol. 15, no. 1, art. 36316, 2025.
31. S. R. Reshma Jebashree and A. M. Anusha Bamini, "LSTM Autoencoder-Based Anomaly Detection for Network Traffic Analysis," in Proceedings of ICOECA, 2026, pp. 1275–1281.
32. D. J. K. Nkashama et al., "ANADOE: Auto-Encoder-Based Network Anomaly Detection with Outlier Exposure," TechRxiv, 2025.
33. S. Alam, Y. Alam, S. Cui, and C. M. Akujuobi, "Unsupervised Network Intrusion Detection Using Convolutional Neural Networks," in Proceedings of IEEE CCWC, 2023, pp. 712–717.
34. K. Wang, "Attention-Based BiGRU-CNN for System Log Anomaly Detection," in Proceedings of ISCTIS, 2024, pp. 151–156.
35. C. Meng and N. Chen, "TinyLog: Log Anomaly Detection with Lightweight Temporal Convolutional Network for Edge Device," in Proceedings of IJCNN, 2024.
36. X. Wei, Q. Liu, and Z. Pan, "Anomaly Detection in Borehole Strain Data with CNN and Frequency-Aware VAE," Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 29, no. 6, pp. 1390–1401, 2025.
37. T. Markovic, S. Moricz, and M. Leon, "Comparative Evaluation of Autoencoders for Semi-Supervised Anomaly Detection on Univariate Time Series Data," in Proceedings of ICMLA, 2024, pp. 1321–1328.
38. N. Borgioli, F. Aromolo, L. T. X. Phan, and G. Buttazzo, "A Convolutional Autoencoder Architecture for Robust Network Intrusion Detection in Embedded Systems," Journal of Systems Architecture, vol. 156, art. 103283, 2024.
39. M. Neri and S. Baldoni, "Unsupervised Network Anomaly Detection with Autoencoders and Traffic Images," in Proceedings of EUSIPCO, 2025, pp. 800–804.
40. S. Casarin, S. Baldoni, M. Carli, P. Zanuttigh, and F. Battisti, "Unsupervised Network Anomaly Detection by Learning on 2D Data Representations," in Proceedings of SDS, 2022, pp. 53–58.
41. L. Ilias et al., "Convolutional Autoencoders Coupled With Hypernetworks for Recognizing Attacks in 5G Networks and Beyond," IEEE Open Journal of the Communications Society, vol. 6, pp. 7885–7898, 2025.
42. J. Gao et al., "MACAE: Memory Module-Assisted Convolutional Autoencoder for Intrusion Detection in IoT Networks," The Journal of Supercomputing, vol. 81, no. 1, art. 231, 2024.
43. D. Drinić, M. Novičić, and G. Kvaščev, "Detection of LR-DDoS Attack Based on Hybrid Neural Networks CNN-LSTM and CNN-Autoencoder," in Proceedings of IcETRAN, 2024.
44. Y. Dai, X. Qian, and C. Yang, "Deep Reinforcement Learning-Based Asymmetric Convolutional Autoencoder for Intrusion Detection," Journal of ICT Standardization, vol. 13, no. 1, pp. 67–82, 2025.
45. M. Ahmed, J. Chen, E. Akpaku, and R. N. A. Sosu, "MTCR-AE: A Multiscale Temporal Convolutional Recurrent Autoencoder for Unsupervised Malicious Network Traffic Detection," Computer Networks, vol. 261, art. 111147, 2025.