Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Direct Current Feature Injection and Adaptive Dynamic Thresholding for Intrusion Detection using Leaky Integrate-and-Fire Neurons

Irshed Hussain
Sri Sivasubramaniya Nadar College of Engineering
Vikash Kumar
vikash.cse@mnit.ac.in

Malaviya National Institute of Technology Jaipur

Research Article

Keywords:

Posted Date: November 12th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11367366/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Direct Current Feature Injection and Adaptive Dynamic Thresholding for Intrusion Detection using Leaky Integrate-and-Fire Neurons

Irshed Hussain[1] and Vikash Kumar[2*†]

[1]Information Technology, Sri Sivasubramaniya Nadar College of Engineering, Chennai, 603110, Tamil Nadu, India.

[2*]Computer Science and Engineering, Malaviya National Institute of Technology Jaipur, Jaipur, 302017, Rajasthan, India.

*Corresponding author(s). E-mail(s): vikash.cse@mnit.ac.in;
Contributing authors: irshedh@ssn.edu.in;
†These authors contributed equally to this work.

Abstract

Intrusion detection systems deployed at the network edge should detect attacks accurately under severe class imbalance while operating within tight energy budgets. Spiking neural networks (SNNs) offer event-driven, low-power, and sparse computation. However, their accuracy is limited by three factors: lossy spike encoding of real-valued traffic features, uncontrolled firing under fixed thresholds, and poorly separable input representations. This paper proposes an SNN-based intrusion detection model that resolves the mentioned limitations. First, a class-aware discriminative autoencoder, trained with a staged schedule of reconstruction pre-training, decoder freezing, and class-weighted fine-tuning, maps PCA-reduced traffic features to compact embeddings that preserve minority-class structure. Second, these embeddings are injected directly as constant membrane currents into leaky integrate-and-fire (LIF) neurons, removing the explicit neural encoding overhead. Third, each LIF neuron uses an adaptive dynamic threshold, driven by its membrane activity and recent spike history, bounded between [0.5, 2.0], which suppresses excessive firing. The network is trained end-to-end with surrogate gradients and backpropagation through time. On NSL-KDD, the proposed model reaches 93.35% accuracy, a macro F1-score of 0.93, and an AUC of 0.966. Compared with a rate-encoded, fixed-threshold SNN baseline, it improves accuracy by 5.2% points, lowers the false-positive rate from 8.4% to 5.1%, and nearly halves missed attacks (13.9% to 7.6%). These results show that direct feature injection combined with adaptive thresholding is a practical route to accurate, energy-aware intrusion detection on neuromorphic and edge hardware.

Keywords: Intrusion detection, NSL-KDD dataset, spiking neural networks, leaky integrate-and-fire neurons, adaptive thresholding, energy efficiency.

1 Introduction

The cyber threat landscape has continued to intensify in recent years, with ransomware emerging as one of the most disruptive forms of cyber-attack. According to the NCC Group Annual

Cyber Threat Intelligence 2025 report, global ransomware attacks increased by 50% year-on-year in 2025, reaching 7,874 reported incidents worldwide [1]. Furthermore, Cybersecurity Ventures estimates that cybercrime costs reached approximately $10.5 trillion annually in 2025, with the cost projected to rise to $12.2 trillion annually by 2031 [2]. Classical defense mechanisms, such as antivirus software and firewalls, often rely on static signatures or predefined rules and may therefore struggle to detect evolving and previously unseen attacks. This makes Intrusion Detection Systems (IDSs) a critical layer of modern cybersecurity frameworks. Unlike traditional signature-based mechanisms, IDSs can identify malicious activities by analyzing network traffic or host behavior. IDSs are broadly categorized into signature-based, anomaly-based, and hybrid approaches, with anomaly-based detection receiving increasing attention because of its ability to identify previously unseen threats.

Recent advances in Machine Learning (ML) and Deep Learning (DL) have further enhanced the capabilities of intelligent intrusion detection systems. Models ranging from decision trees and Support Vector Machines (SVMs) to deep architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks have demonstrated promising performance on benchmark datasets, including NSL-KDD [3], UNSW-NB15 [4], and CIC-IDS2017. The major strengths of these approaches include their ability to learn complex patterns, adapt to diverse traffic characteristics, and process high-dimensional network data. Despite their effectiveness, conventional DL models can involve computationally intensive architectures and substantial memory and data requirements, which can limit their suitability for real-time and resource-constrained cybersecurity environments.

Spiking Neural Networks (SNNs) provide an alternative computational paradigm inspired by biological neural systems, in which information is represented and processed through discrete spikes. Their event-driven operation and sparse computation offer potential advantages in terms of latency and energy efficiency, making SNNs attractive for resource-constrained and neuromorphic security applications [5]. SNN-based approaches have consequently been explored for intrusion and anomaly detection. For example in [6], an unsupervised SNN-based anomaly detection approach that does not require labeled data and is designed for dynamic environments in which data distributions evolve over time, is proposed.

However, the effectiveness of SNN-based intrusion detection is constrained by several challenges. First, conventional spike-encoding schemes transform continuous-valued network features into discrete spike trains, which can discard information contained in the original feature magnitudes and introduce additional temporal processing and latency. Second, fixed neuronal firing thresholds may lead to excessive or insufficient spiking when the input characteristics vary substantially across features or classes, thereby limiting the ability of the network to represent discriminative patterns effectively. Third, SNNs remain highly dependent on the separability of their input representations. Local learning mechanisms such as Spike-Timing-Dependent Plasticity (STDP) can adapt synaptic weights efficiently, but their effectiveness may deteriorate when input classes exhibit substantial overlap or severe class imbalance [7]. Importantly, dimensionality reduction techniques such as Principal Component Analysis (PCA), although effective for retaining dominant variance in high-dimensional network traffic, do not explicitly optimize class separability. Consequently, preserving the principal directions of variance does not necessarily produce representations that are well suited for distinguishing minority attack classes from dominant traffic patterns.

To address these limitations, this work proposes a class-aware representation learning framework that combines nonlinear feature learning with direct-current SNN processing and adaptive neuronal thresholds. First, PCA is employed as an initial dimensionality-reduction stage to remove redundancy and retain the dominant information in the network traffic features. The resulting representation is then refined using a Class-Aware Denoising Autoencoder (CA-DAE), which jointly optimizes reconstruction and class-weighted discrimination objectives. Unlike PCA, the CA-DAE explicitly incorporates class information during representation learning, enabling it to learn compact nonlinear embeddings with improved minority-class separability under imbalanced and overlapping traffic distributions. In the

first model (Model-1), these embeddings are provided to an SNN through conventional spike generation. In Model-2, the learned continuous embeddings are instead injected directly as neuronal input currents, eliminating explicit spike encoding and its associated computational overhead. Furthermore, Model-2 incorporates an adaptive neuronal threshold that dynamically regulates spike generation according to the neuron's activity, replacing the fixed threshold with a time-varying threshold. This combination of class-aware representation learning, direct-current feature injection, and adaptive thresholding enables the SNN to operate on more discriminative representations while reducing information loss and regulating neuronal firing.

The main contributions of this work are:

1. A class-aware discriminative autoencoder (CA-DAE) trained with a staged schedule – reconstruction pre-training, decoder freezing, and class-weighted fine-tuning that produces compact embeddings in which minority attack classes remain separable, giving the SNN a well-conditioned input space.
2. Direct-current feature injection, in which the CA-DAE embeddings drive LIF membranes as constant input currents, eliminating explicit neural encoding and its associated computational overhead while enabling end-to-end intrusion classification.
3. A bounded adaptive dynamic threshold denoted by $\theta^{(i)}(t)$ as $\theta^{(i)}(t) = \text{clip}\left[\theta_0 + \alpha|\bar{V}_m^{(i)}(t)| + \beta A^{(i)}(t), \theta_{\min}, \theta_{\max}\right]$ that couples each LIF neuron's threshold to its membrane activity and recent firing, regulating excessive firing and potentially improving feature discrimination and classification performance.
4. A systematic evaluation on NSL-KDD, including an ablation that isolates each component, an operation-count-based energy analysis, and comparisons with a rate-encoded fixed-threshold baseline SNN model and an adaptive dynamic threshold based SNN model with direct current injection.

The remainder of the paper is structured as follows: Section 2 reviews related work in SNN-based learning and intrusion detection. Section 3 presents the proposed class-aware DAE for imbalance-robust nonlinear representation learning. The proposed models are discussed in detail in sections 4 & 5. Section 6 details the experimental setup and results. Finally, section 7 concludes with key insights and future directions.

2 Related Work

Spiking neural networks have been applied to intrusion detection both as stand-alone classifiers and as components of hybrid deep architectures. This section reviews these works under three themes – hybrid and deep SNN-based IDS (Section 2.1), input encoding and neuron-level design (Section 2.2), and energy-efficient edge deployment (Section 2.3) – and then states the research gap addressed in this paper (Section 2.4).

2.1 Hybrid and deep SNN-based IDS

Wang *et al.* [8] proposed a convolutional SNN that improves detection performance while reducing inference complexity, although the convolutional front end increases architectural complexity and limits interpretability. Xie and Chen [9] showed that convolutional SNNs can model temporal traffic patterns, but their model depends on careful spike encoding and is less robust under highly variable network conditions. Karthik *et al.* [10] combined attention with spiking dynamics in a protocol-aware Transformer–spiking model for energy-efficient detection; the work targets overall accuracy and does not address class imbalance or minority-class representation. Zarzoor *et al.* [11] used SNN-based feature extraction with a decision-tree classifier; because the features are pre-selected and handcrafted, the model generalises poorly to unseen attack types. Mustafa *et al.* [12] coupled a recurrent network with an SNN to capture temporal dependencies in IoT traffic, retaining a dense recurrent component alongside the spiking stage. Muthusenthil and Devi [13] integrated deep convolutional SNNs with blockchain for privacy-preserving intrusion detection in cloud environments; this framework focuses on securing the detection pipeline and relies on deep convolutional stacks.

In these works, performance gains come mainly from adding ANN components – convolution, attention, recurrence or decision trees –around a

conventional spiking core. How inputs enter the spiking neurons, and when those neurons fire, receives comparatively little attention.

2.2 Input encoding and neuron-level design

At the neuron level, Prajwalasimha *et al.* [14] employed STDP-based event-driven learning for edge and IoT security, but unsupervised local plasticity is sensitive to feature overlap and training stability. Gretsch *et al.* [15] proposed single-spike networks that reduce computational cost by restricting each neuron to one spike, which limits representational richness for complex multi-class traffic. Related streaming approaches adapt SNNs online to concept drift [16] or through semi-supervised lifelong learning [17], which is complementary to the offline supervised setting studied here.

Outside intrusion detection, two lines of work bear directly on this paper. The first concerns input encoding. Rate coding converts each real-valued feature into a stochastic spike train and needs many time steps to represent values accurately, whereas direct coding feeds real values to the first layer and lets that layer generate spikes [18]. Kim *et al.* [19] showed that direct coding reaches higher accuracy with far fewer time steps than rate coding. The second concerns threshold dynamics. Adaptive LIF (ALIF) neurons raise their threshold after each spike and let it relax afterwards, which regulates firing and extends temporal memory [20, 21], while parametric LIF neurons learn their membrane time constants during training [22]. These mechanisms have been developed and evaluated mainly on vision and speech benchmarks, and in ALIF neurons the threshold depends only on the neuron's own spike history. To the best of our knowledge, neither direct coding nor adaptive thresholds have been examined for network intrusion detection, where inputs are tabular, heterogeneous and severely imbalanced.

2.3 Energy-efficient and edge SNN-based IDS

Dinler [23] proposed a task-scaled SNN with multi-stage feature selection for resource-constrained IoT devices; dynamic feature selection improves efficiency, but the reliance on selected features limits representation learning and the class imbalance problem is not addressed. Živadinović and Simić [24] showed that SNNs can substantially reduce energy consumption for IoT intrusion detection, but reported that this reduction comes at the cost of detection performance, particularly with imbalanced data and evolving attacks. Together, these studies confirm the energy case for SNN-based IDS, but also show that efficiency is often gained at the expense of minority-class detection.

2.4 Research gap

A review of state-of-the-art studies in this field reveals three recurring gaps in SNN-based intrusion detection. (i) *Input representation:* most models operate on raw or handcrafted features, or depend on ANN front ends, rather than on compact representations explicitly shaped for class separability under imbalance. (ii) *Encoding:* features are converted to spikes by rate or temporal coding, which adds latency and information loss, and direct current input—effective in other domains [18, 19]—has not been examined for intrusion detection. (iii) *Neuron dynamics:* the reviewed SNN-IDS use fixed firing thresholds, and the adaptive thresholds proposed elsewhere respond only to spike history, so firing is not regulated with respect to membrane activity as input statistics shift.

This paper addresses all the aforementioned gaps within a single framework: a class-aware discriminative autoencoder with staged optimisation that produces separable, imbalance-robust embeddings (Section 3); direct current injection of these embeddings into LIF neurons, removing the explicit spike-encoding stage (Section 5.1); and a bounded adaptive threshold that responds to both membrane activity and recent firing (Section 5.2). Table 1 summarizes the key mathematical symbols and variables used in Algorithms 1, and 2.

3 Class-Aware DAE for Robust Intrusion Representation

To build and evaluate the IDS, we use the benchmark NSL-KDD dataset [3], which contains 41 network traffic features, an attack label, and a difficulty score. Irrelevant features, such as the difficulty score and service, are removed during

Table 1 Summary of notations used in the proposed DAE and SNN framework.

Symbol	Description	Value
$\hat{X}$	Reconstructed input from the decoder	–
$\hat{Y}$	Predicted class output	–
$\mathcal{L}_{rec}$	Reconstruction loss	–
$\mathcal{L}_{cls}$	Weighted classification loss	–
$\mathcal{L}_{total}$	Total training loss	–
E	Total number of training epochs	–
P	Early stopping patience	–
T	Number of simulation time steps	–
$\theta = \{W_l, b_l\}$	Synaptic weights and biases of layer l	–
$\mathcal{L}$	Weighted cross-entropy loss	–
C	Total number of classes	–
$y \in \{0, \ldots, C-1\}$	Encoded class label	–
$\mathcal{D}_{train}, \mathcal{D}_{val}$	Training and validation datasets	–
$\mathcal{L}_{train}, \mathcal{L}_{val}$	Training and validation data loaders	–
$V_l(t)$	Membrane potential of layer l at time t	–
$S_l(t)$	Output spike of layer l at time t	–
∇_θ	Gradient-based parameter update	–
$\theta^{(i)}(t)$	Dynamic threshold of neuron i at time t	–
θ_0	Initial threshold	1.0
α	Sensitivity to membrane activity	0.20
$\bar{V}_m^{(i)}(t)$	Membrane potential for threshold adaptation	–
β	Sensitivity to recent firing activity	0.50
$A^{(i)}(t)$	Recent spike-history adaptation state	–
ρ	Decay factor of firing history	0.90
$S^{(i)}(t)$	Spike emitted by neuron i	–
$V_m^{(i)}(t)$	Membrane potential of neuron i	–
$H(\cdot)$	Heaviside spike-generation function	–
$\theta_{\min}$	Minimum threshold	0.5
$\theta_{\max}$	Maximum threshold	2.0

preprocessing. The attack types are grouped into five classes: *Normal, DoS, Probe, R2L*, and *U2R*. Missing values are handled using median imputation, followed by a stratified train-test split to preserve the empirical class distribution and reduce sampling bias. To address severe class imbalance, SMOTE is applied to the training data to oversample minority classes [25]. Finally, the features are standardized using z-score normalization to reduce scale-related bias and ensure scale invariance across the feature space.

After performing oversampling and standardization, the proposed method employs a class-aware DAE to model the nonlinear feature representations. The main goal of DAE is to optimize reconstruction and classification objectives as follows:

$$\mathcal{L}_{\text{DAE}} = \mathcal{L}_{\text{rec}}(\mathbf{X}, \hat{\mathbf{X}}) + \lambda\, \mathcal{L}_{\text{cls}}(y, \hat{y}) \tag{1}$$

where $\mathcal{L}_{\text{rec}}$ enforces reconstruction fidelity and $\mathcal{L}_{\text{cls}}$ represents supervised classification loss.

The problem with the baseline configuration is that the optimization is mostly dominated by the reconstruction objective. Therefore, in case of severe class imbalance, the learned representation results in bias toward majority traffic patterns. This results in representation collapse for minority classes (*R2L* and *U2R*), and consequently limits the accuracy to approximately 50%. To mitigate this limitation, a class-weighted supervision is introduced as:

$$\mathcal{L}_{\text{cls}} = -\sum_{c=1}^{C} w_c\, y_c \log(\hat{y}_c), \tag{2}$$

where w_c is inversely proportional to the frequency of class c. This weighting amplifies gradient contributions from minority classes and improves inter-class separation in the latent space. With staged optimization and decoder freezing, the DAE achieves an accuracy of 94.92% which is close to PCA performance while providing class-aware nonlinear embeddings. To stabilize training under severe class imbalance, a staged optimization strategy is adopted which involves extended reconstruction pretraining followed by decoder freezing and discriminative fine-tuning with class-weighted supervision. Figure 1 presents a comparative t-SNE visualization of PCA-reduced features ($n = 10$) and tuned DAE latent representations. It can be observed from the figure that the PCA-based embedding produces compact, globally coherent clusters for the dominant traffic categories, with smoother inter-class boundaries and reduced overlap.

In contrast, the tuned DAE reveals pronounced local nonlinear structure, with compact and semantically coherent manifolds for high-frequency attack classes such as DoS and Probe. Compared with PCA, it produces more fragmented yet class-aware clusters, while minority classes such as R2L and U2R remain smaller and sparser but show improved separation. Staged optimization, decoder freezing, and class-weighted supervision help mitigate representation collapse

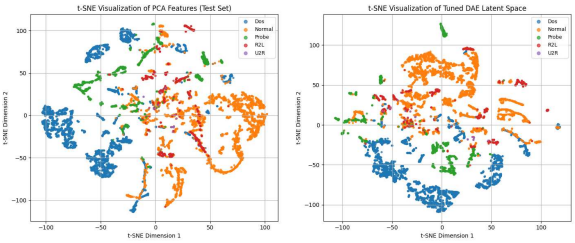


Figure 1 t-SNE comparison of PCA-reduced features and tuned DAE latent representations on the test set.

under severe class imbalance, producing compact, class-aware embeddings $\mathbf{z} \in \mathbb{R}^d$ suitable for spike-based encoding. These embeddings can be converted into spike trains using rate- or latency-based coding,

$$r_i \propto z_i \quad \text{or} \quad t_i \propto \frac{1}{z_i}, \tag{3}$$

thereby addressing the sample-overlap issue discussed in Section 1 and enabling energy-efficient, event-driven SNN-based intrusion detection.

4 Development of Proposed Model-1

Two model variants – Model-1 and Model-2 is proposed. Model-1 serves as a baseline, and Model-2 addresses its limitations, improving overall performance. The block diagram in Figure 2 shows the main components of the proposed Model-1 including preprocessing followed by feature extraction. The next step is the neural encoding (rate encoding) to convert real-valued inputs to spikes; then the SNN architecture, composed of synapse and neuron models. The proposed Model-1 parameters were tuned using the Adam optimizer and surrogate gradient learning [26], implemented with Backpropagation Through Time (BPTT), to ensure stable training and reliable performance evaluation.

4.1 Neural Encoding

The working of rate encoding is as follows: let the input vector be $\mathbf{x} \in \mathbb{R}^d$, where d is the input dimensionality. The input is injected at each discrete simulation step $t \in \{1, \ldots, T\}$. The encoded input current to neuron j passed by input layer neurons

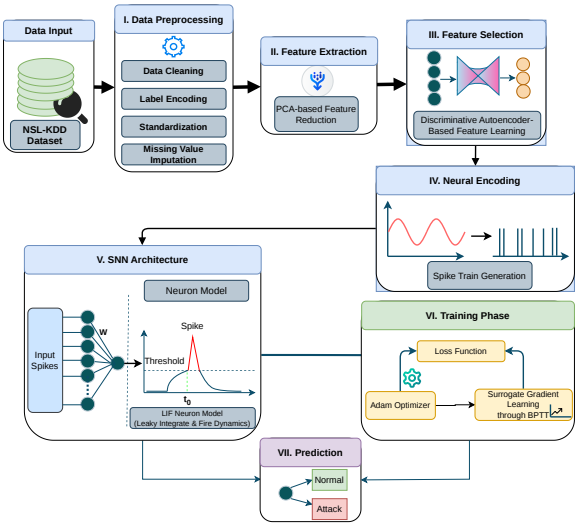


Figure 2 The block diagram that clearly represents the overall procedure followed for the development of Model-1. Each essential components from data collection to prediction is lucid from the block diagram.

in the first hidden layer is calculated using Eq. (4).

$$I_j^{(1)}(t) = \sum_{i=1}^{d} w_{ji}^{(1)} x_i \tag{4}$$

where $w_{ji}^{(1)}$ denotes the synaptic weights. Neuronal membrane evolution with time is the source of temporal dynamics in rate encoding. Figure 3 shows the overall method to transform the real-valued features into spikes using rate encoding method.

4.2 Neuron Model

Each neuron used in the hidden and output layers are the Leaky Integrate-and-Fire (LIF) [27, 28] spiking neuron model. The working of LIF neuron is as follows: let us consider $V_m(t)$ is the membrane potential at time t. Then the membrane potential evolves as described in Eq. (5).

$$\frac{dV_m(t)}{dt} = -\frac{1}{\tau_m}\left(V_m(t) - V_{\text{rest}}\right) + \frac{R}{\tau_m} I(t) \tag{5}$$

where τ_m denotes the time constant, V_{rest} is the resting membrane potential, R is the resistance, and $I(t)$ represents the input current at time t. In

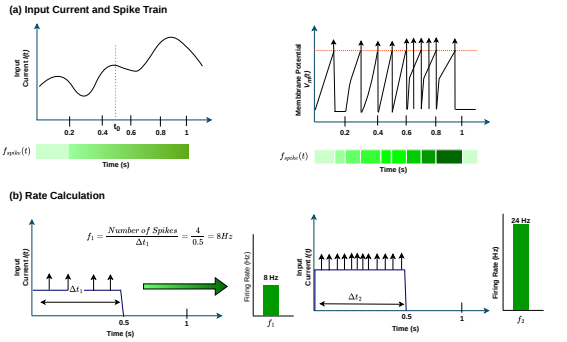


Figure 3 Illustration of the method for the conversion of real-valued features into spikes (a) Input current versus time and the change in membrane potential V_m with respect to time t is shown where there is spike once $V_m(t)$ reaches a defined threshold (b) Calculation of spike firing rate f_1 where number of spikes are getting divide by a time reference window Δt_1. After calculation f_1 is found to be 8 Hz, and f_2 is 24 Hz.

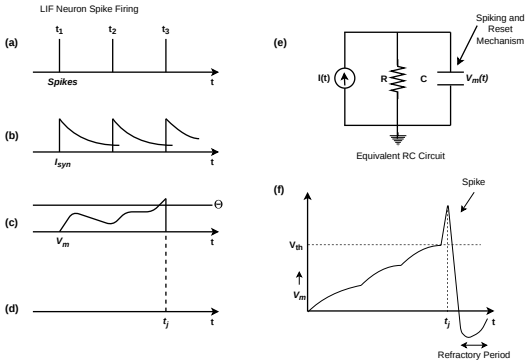


Figure 4 Illustration of the behaviour and working principle of the LIF spiking neuron (a) Three discrete events t_1, t_2, t_3 in the form of spikes are shown with the same amplitude (b) Spikes are represented in the form of synaptic current I_{syn} (c) The integration of the total synaptic current received by a post-synaptic neuron and when the membrane potential V_m reaches a defined threshold Θ, then there is a spike fired (d) The value of V_m reaches threshold Θ at time t_j.

our case, we set the value $V_{rest} = 0$. The value of $I(t)$ is calculated using Eq. (6).

$$I(t) = \sum_i w_i \sum_k \alpha\big(t - t_i^{(k)}\big) \tag{6}$$

where w_i denotes the weight of presynaptic neuron i, $t_i^{(k)}$ is the k^{th} spike of neuron i, and $\alpha(t)$ denotes the synaptic response kernel. When V_m crosses the threshold Θ at time t, there is a spike fired at time t. Figure 4 shows the behaviour and working technique for LIF spiking neuron. In addition, the equivalent RC circuit for the corresponding LIF neuron is also shown with spiking and reset mechanism, where $I(t)$ is the input current, R is the resistance, C is the capacitance, and $V_m(t)$ is the potential. A spike is generated when $V_m(t)$ crosses $\Theta \in \mathbb{R}$. The spike train $s(t)$ of the neuron can be formally written as in Eq. (7), where $H(\cdot)$ is the Heaviside step function.

$$s(t) = H\big(V_m(t) - \Theta\big) \tag{7}$$

4.3 Proposed Network Topology

A fully connected feedforward network is proposed for the Model-1 which is shown in Figure 5. In the architecture, there are a total of 39 input neurons, two hidden layers (trial and error), first hidden layer consists of 64 hidden neurons, second hidden layer consists of 32 hidden neurons, and 2 output neurons.

4.4 Training with Surrogate Gradient and BPTT

The non-differentiability of the spike generation function poses a major obstacle for gradient-based learning. Particularly, the derivative of the Heaviside function is zero almost everywhere, making direct gradient propagation infeasible. To overcome this limitation, surrogate gradient [26] methods replace true derivative of the spike function during the backward pass with a smooth, non-zero surrogate, while preserving the exact spiking dynamics during the forward pass. This approximation enables effective credit assignment across layers and time, allowing deep and recurrent SNNs to be trained using gradient descent [29, 30]. In the discrete-time setting, the spike derivative with respect to the membrane potential is approximated as shown in Eq. (8).

$$\frac{\partial s_t}{\partial V_m(t)} \approx \sigma'\big(V_m(t) - \Theta\big) \tag{8}$$

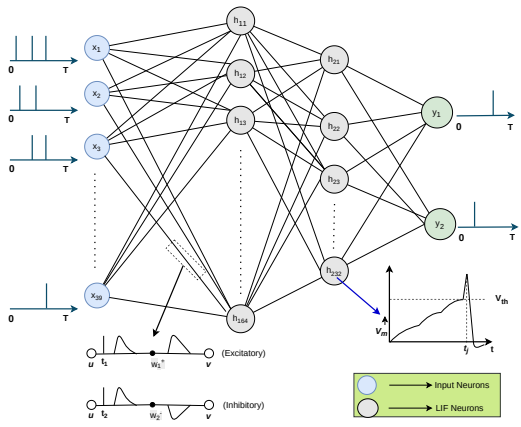


Figure 5 The proposed feedforward network architecture for Model-1, where there are a total of 39 input neurons, two hidden layers, first hidden layer consists of 64 hidden neurons, second hidden layer consists of 32 hidden neurons, and 2 output neurons. Input neurons are represented using $x_1, x_2, \ldots, x_{39}$, hidden neurons are represented using $h_{11}, h_{12}, \ldots, h_{164}$. The meaning of h_{11} is first hidden neuron of first hidden layer, similarly the meaning of h_{232} is 32^{th} hidden neuron of second hidden layer.

where $\sigma'(\cdot)$ is a smooth surrogate derivative, commonly chosen as given in Eq. (9).

$$\sigma'(x) = \frac{1}{(1+|x|)^2} \tag{9}$$

The discretized membrane dynamics can be written as given in Eq. (10).

$$V_m(t) = \alpha V_m(t-1) + \sum_i w_i x_i(t) - \Theta s_{t-1} \tag{10}$$

where $\alpha = e^{-\Delta t/\tau_m}$ and $x_i(t)$ denotes filtered presynaptic input.

To compute gradients, BPTT is applied. Let us consider $\delta_t = \frac{\partial \mathcal{L}}{\partial V_m(t)}$ denote the error signal at time t. It satisfies the recursive relation given in Eq. (11).

$$\begin{aligned}\delta_t = \frac{\partial \mathcal{L}}{\partial s_t} \cdot \sigma'\big(V_m(t) - \Theta\big) \\ + \delta_{t+1}\left(\alpha - \Theta \cdot \sigma'\big(V_m(t) - \Theta\big)\right)\end{aligned} \tag{11}$$

The first term captures the instantaneous contribution of time step t to the loss, while the second term propagates error information backward from future time steps due to temporal dependencies in the membrane dynamics.

For the MSE loss function, the gradient with respect to the spike output is given by Eq. (12).

$$\frac{\partial \mathcal{L}}{\partial s_t} = \frac{2}{T}(s_t - y_t) \tag{12}$$

The gradient of the loss with respect to each synaptic weight w_i is then obtained by accumulating contributions over time using Eq. (13).

$$\frac{\partial \mathcal{L}}{\partial w_i} = \sum_{t=1}^{T} \delta_t \, x_i(t) \tag{13}$$

where $x_i(t)$ denotes the filtered input from presynaptic neuron i.

4.4.1 Surrogate Gradient Approximation

The temporal dependency of the membrane potential introduces a recursive gradient term given in (14).

$$\frac{\partial \mathcal{L}}{\partial {V_m}_i^t} = \frac{\partial \mathcal{L}}{\partial s_i^t}\frac{\partial s_i^t}{\partial {V_m}_i^t} + \lambda \frac{\partial \mathcal{L}}{\partial {V_m}_i^{t+1}} \tag{14}$$

which propagates errors backward in time. However, since $\partial s_i^t / \partial {V_m}_i^t = 0$ almost everywhere, the true gradient vanishes. To enable learning, surrogate gradient methods replace this term during backpropagation with a smooth approximation $\sigma'({V_m}_i^t - \Theta)$, resulting in as shown in Eq. (15).

$$\frac{\partial \mathcal{L}}{\partial {V_m}_i^t} \approx \frac{\partial \mathcal{L}}{\partial s_i^t}\sigma'({V_m}_i^t - \Theta) + \lambda \frac{\partial \mathcal{L}}{\partial {V_m}_i^{t+1}} \tag{15}$$

This approximation makes BPTT applicable to LIF neurons and enables efficient gradient-based training of SNNs despite the non-differentiability of spike events [26, 30]. After computing the gradients, the parameters are updated using the Adam optimization algorithm [31]. Adam adapts the learning rate for each parameter by maintaining estimates of both the mean and variance of the gradients, which improves convergence and stability in training [31].

5 Development of Proposed Model-2

The Model-2 is an improvement over Model-1 where the LIF neuron takes the directly injected current as input rather than the encoded spikes. For Model-2, the network topology and the training algorithm is same as Model-1. Limitations of Model-1 are as follows: it uses a constant-current encoding method (rate encoding), and an MSE loss function on spike counts. To overcome the issues encountered in Model-1, Model-2 was developed and successfully tested on the same dataset. Like Model-1, Model-2 also uses rigorous preprocessing and feature engineering methods. Since there is no standard neural encoding scheme that works best for SNN. Therefore, in Model-2, instead of rate encoding, the direct current injection method is used, resulting in a significant increase in classification accuracy. *Model-2 also replaces Model-1's fixed threshold with an adaptive dynamic threshold, regulating excessive firing and potentially improving feature discrimination and classification performance of Model-1.*

5.1 Neural Encoding

In Model-2, instead of converting input data into spike trains using an encoding scheme, the input is applied directly as a constant current to the neuron's membrane. This approach is often used in theoretical studies and hardware-oriented research because it isolates the intrinsic dynamics of spiking neurons without introducing additional encoding complexity. Direct current injection refers to feeding the raw input value as a constant external current I_{ext} into the membrane equation of a spiking neuron. The neuron then converts this continuous current into spikes through its natural integrate-and-fire dynamics. Mathematically, the input signal x is mapped directly as $I_{ext} = k.x$, where x is the input feature value, k is the scaling factor (normalization or gain), I_{ext} is the injected external current. The neuron generates spikes depending on whether the integrated membrane potential crosses the firing threshold.

Direct current injection is typically described within neuron models such as the LIF model, which approximates biological neuron membrane dynamics. The membrane potential evolution is given in Eq. (16).

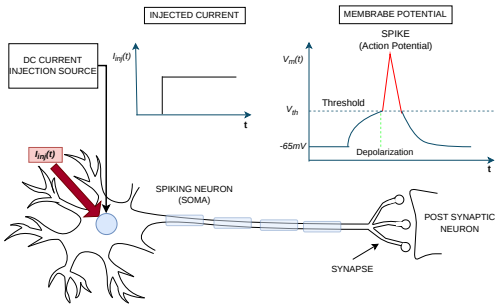


Figure 6 Illustration of DC current injection $I_{inj}(t)$ at source at time t, where instead of input encoding direct current injection is done which act as input stimuli for a post-synaptic neuron. After receiving the input stimuli from the pre-synaptic neurons, the membrane potential $V_m(t)$ raises and reaches a defined threshold V_{th} to generate a spike. The time at which $V_m(t)$ reaches V_{th} is recorded.

$$\tau_m \frac{dV(t)}{dt} = -(V_m(t) - V_{rest}) + RI_{inj}(t) \quad (16)$$

Where, $V(t)$ is the potential of the membrane at time t, V_{rest} is the resting potential, τ_m is the time constant, R is the membrane resistance, $I_{inj}(t)$ is the directly injected input current. Thus, spike generation frequency is proportional to the magnitude of injected current. Figure 6 shows how direct current injection is done as the input stimuli for a post-synaptic neuron.

5.2 Proposed Neuron Model

Although same LIF neuron is used in Model-2 as used in Model-1. However, Model-2 replaces Model-1's fixed threshold in spike generation function $S^{(i)}(t) = H(V_m^{(i)}(t) - 1)$ with an adaptive dynamic threshold $S^{(i)}(t) = H(V_m^{(i)}(t) - \theta^{(i)}(t))$ for i^{th} LIF neuron, where $\theta^{(i)}(t)$ is described using Eq. (17). The meaning and interpretation of the symbols used in Eq. (17) is described in Table 1. It regulates excessive firing and potentially improving feature discrimination and classification performance of Model-1.

$$\theta^{(i)}(t) = \text{clip}\Big[\theta_0 + \alpha|\bar{V_m^{(i)}}(t)| + \beta A^{(i)}(t), \theta_{\min}, \theta_{\max}\Big] \quad (17)$$

5.3 Loss Function

Cross entropy is used as loss function unlike Model-1 that used MSE. The loss function is described using Eq. (18).

$$L = -\sum_{i=1}^{C} y_i \log \hat{p}_i \tag{18}$$

Where, $\hat{p}_i$ is computed using Eq. (19).

$$\hat{p}_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{19}$$

The proposed Model-2 is described clearly with the Algorithm 1, and Algorithm 2. The procedure LIF_DYNAMICS(.) takes the input stimuli and computes the output spikes according to the dynamics of LIF neuron as described in Eq. (16).

Algorithm 1 FORWARD_SNN(X, θ, T)

Require: $X \in \mathbb{R}^d, \theta = \{W_l, b_l\}, T$
Ensure: $\hat{Y}$

Initialize $V_l := 0$ ▷ Resetting neuronal potentials for layer l
$out := 0$ ▷ Accumulate output spikes
for $t = 1$ to T **do**
  $S_1(t) :=$ LIF_DYNAMICS$(X \times W_1 + b_1)$ ▷ 1^{st} hidden layer spikes
  $S_2(t) :=$ LIF_DYNAMICS$(S_1(t) \times W_2 + b_2)$ ▷ 2^{nd} hidden layer spikes
  $S_3(t) :=$ LIF_DYNAMICS$(S_2(t) \times W_3 + b_3)$ ▷ Output layer spikes
  $out := out + S_3(t)$ ▷ Temporal accumulation
end for
$\hat{Y} \leftarrow \frac{1}{T} \sum_{t=1}^{T} S_3(t)$
return $\hat{Y}$

5.4 Energy Efficiency of Model-1 and Model-2

One of the primary motivations for SNN models is the energy-efficiency. In this section, the energy efficiency of ANN and SNN is theoritically analysed. Let us consider E_{MAC} represents the energy cost of a Multiply Accumulate (MAC) operation, E_{AC} represents the energy cost of an Accumulate

Algorithm 2 Proposed Model Training with Early Stopping

Require: X, E, P, T
Ensure: Optimized parameters θ^*

Initialize synaptic weights θ
Define weighted loss: $\mathcal{L} = -\sum_{c=0}^{C-1} w_c \, y_c \log(\hat{y}_c)$
$best := \infty$, $counter := 0$ ▷ Early stopping variables
for $i = 1$ to E **do**
  for each $(X_b,\ y_b) \in \mathcal{L}_{train}$ **do**
    $\hat{Y}_b :=$ FORWARD_SNN(X_b, θ, T) ▷ From Algorithm 1
    Calculate the loss $\mathcal{L}(\hat{Y}_b, y_b)$
    Update θ using $\nabla_\theta \mathcal{L}$ ▷ Backpropagation with surrogate gradients
  end for
  for each $(X_b, y_b) \in \mathcal{L}_{val}$ **do**
    $\ell_b = \mathcal{L}($FORWARD_SNN$(X_b, \theta, T), y_b)$
  end for
  $\mathcal{L}_{val} = \frac{1}{M} \sum_b \ell_b$
  if $\mathcal{L}_{val} < best$ **then**
    $best := \mathcal{L}_{val}$, save θ^* ▷ Keeping the best
    $counter := 0$
  else
    $counter := counter + 1$
    if $counter \geq P$ **then**
      break ▷ If no improvement, stop
    end if
  end if
end for
return θ^*

(AC) operation, N is the number of neurons, M is the number of synapses, T is the number of discrete time steps, and $r \in [0, 1]$ is the average firing rate of neurons. Empirically, on digital hardware, $E_{\text{MAC}} \gg E_{\text{AC}}$ often by one to two orders of magnitude.

In a feedforward ANN, each neuron calculates a weighted sum then passes through a nonlinear activation as given in the Eq. (20). The activation function is denoted by $\phi(.)$, and O_j is the output produced by a neuron j.

$$O_j = \phi\left(\sum_{i=1}^{M} w_{ji} x_i\right) \tag{20}$$

Where, each synapse requires one MAC operation. Therefore, the total energy per inference is $E_{\text{ANN}} = M \cdot E_{\text{MAC}}$.

However, in an SNN, synaptic computation occurs only when a presynaptic spike is emitted. Thus, the synaptic current denoted by $I_j(t)$ is given in Eq. (21), where $s_i(t) \in \{0, 1\}$.

$$I_j(t) = \sum_i w_{ji} s_i(t) \tag{21}$$

Over T time steps, the expected number of synaptic events is given by $\mathbb{E}[\text{Synaptic Ops}] = r \times M \times T$. Each synaptic event requires only an accumulate operation i.e., $E_{\text{SNN, syn}} = rMT \cdot E_{\text{AC}}$. Thus, SNNs achieve energy efficiency by replacing MAC operations with AC operations [32]. In addition, membrane updates introduce a small overhead given by $E_{\text{mem}} = NT \cdot E_{\text{AC}}$. Therefore, the total energy is calculated using Eq. (22).

$$E_{\text{SNN}} = rMTE_{\text{AC}} + NTE_{\text{AC}} \tag{22}$$

For large networks where $M \gg N$, therefore $E_{\text{SNN}} \approx rMTE_{\text{AC}}$.

The energy ratio between SNN and ANN inference is calculated using Eq. (23).

$$\frac{E_{\text{SNN}}}{E_{\text{ANN}}} \approx \frac{rTE_{\text{AC}}}{E_{\text{MAC}}} \tag{23}$$

Since $E_{\text{MAC}} \approx 10 - 100 \times E_{\text{AC}}$, the ratio is obtained using Eq. (24).

$$\frac{E_{\text{SNN}}}{E_{\text{ANN}}} \approx \frac{rT}{10 - 100} \tag{24}$$

Thus, SNNs are more energy efficient when $rT \ll 10$. It highlights the core trade-off among the low firing rates ($r \ll 1$), moderate simulation time (T), and the sparse event-driven activity. Under these conditions, SNNs can achieve one to two orders of magnitude lower energy consumption than ANNs. In addition, ANNs perform dense computation: $\forall i, j$: compute $w_{ji}x_i$, while SNNs perform sparse computation: compute $w_{ji} \iff s_i(t) = 1$. It replaces spatial density with temporal sparsity, and thus converting energy cost from: $\mathcal{O}(M)$ to $\mathcal{O}(rMT)$.

Table 2 Experimental setup and hyper-parameters.

Parameter	Model-1	Model-2
Network topology	39–64–32–2	39–64–32–2
Input to first layer	Rate encoding	Direct current injection
Firing threshold	Fixed, $\Theta = 1.0$	Adaptive, $\theta_0 = 1.0$, bounds [0.5, 2.0]
Threshold sensitivities α, β	–	0.20, 0.50
Firing-history decay ρ	–	0.90
Surrogate derivative	$1/(1 + \lvert x \rvert)^2$	$1/(1 + \lvert x \rvert)^2$
Loss function	MSE	Weighted cross-entropy
Simulation time steps T	–	5
Optimizer	Adam	Adam
Training epochs	500	Early stopping (convergence at ≈325)

6 Experimental Results and Discussion

This section reports the results of Model-1, the rate-encoded, fixed-threshold baseline, and Model-2, which adds direct current injection and the adaptive dynamic threshold.

6.1 Experimental setup

Table 2 lists the hyper-parameters used for both models. Results are reported on the held-out test set of 51,535 samples for binary classification, where Class 0 denotes normal traffic and Class 1 denotes intrusion attacks.

6.2 Experimental Results for Model-1

The accuracy curve of Model-1 in Figure 7 rises quickly in the first few epochs and then fluctuates between approximately 85% and 90% for the remaining epochs. The close alignment between training and test accuracy indicates that the model does not suffer from significant overfitting; however, the large fluctuations show that training with the MSE loss on spike outputs is unstable, which is one of the reasons for adopting the cross-entropy loss in Model-2.

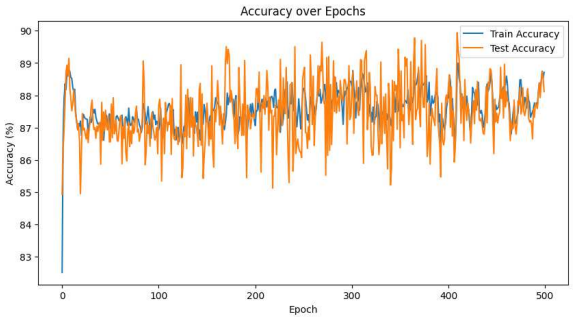


Figure 7 Training and test accuracy versus epochs for Model-1.

In the confusion matrix of Figure 8, Model-1 correctly classifies 17,021 normal and 28,389 attack samples, while 1,555 normal samples are misclassified as attacks and 4,570 attacks are misclassified as normal traffic.

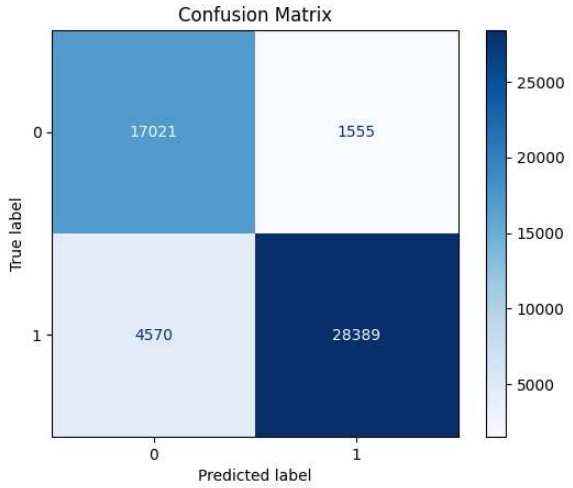


Figure 8 Confusion matrix for binary classification, where Class 0 denotes normal traffic and Class 1 denotes intrusion attacks, for Model-1.

The class-wise metrics computed from this confusion matrix are given in Table 3. Model-1 reaches an overall accuracy of 88.11%. The precision for intrusion attacks, i.e., Class 1 (Attack), is high (0.95), but the recall for this class is 0.86, which means that 13.87% of the attacks are missed. For normal traffic, i.e., Class 0 (Normal), the recall is 0.92, while the precision is only 0.79, and 8.37% of normal samples are raised as false alarms. The number of missed attacks is therefore the main weakness of Model-1, and these are the limitations that Model-2 addresses.

Table 3 Class-wise performance comparison of Model-1 across different classes.

Class	Prec.	Recall	F1	Acc. (%)
Class 0 (Normal)	0.79	0.92	0.85	88.11
Class 1 (Attack)	0.95	0.86	0.90	

Figure 9 presents the membrane potential of the two output neurons of Model-1 for a single sample over time steps. The membrane potential of Neuron 1 increases steadily as synaptic inputs are integrated, while that of Neuron 0 remains low, so the sample is assigned to Class 1. This confirms correct temporal integration of the input within the SNN.

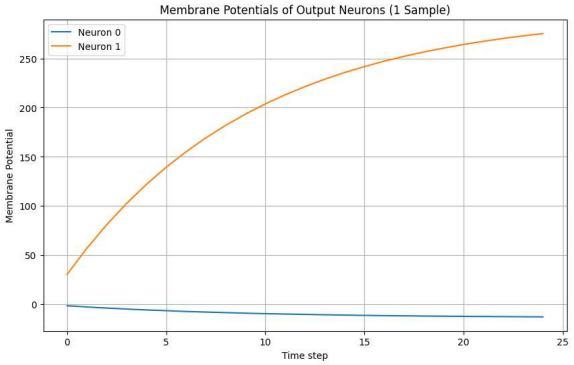


Figure 9 Membrane potential of the output neurons of Model-1 versus time steps for a single sample.

6.3 Experimental Results for Model-2

The accuracy curve of Model-2 (Figure 10) converges at around 325 epochs and is considerably smoother than that of Model-1.

In the confusion matrix of Figure 11, Model-2 correctly classifies 17,660 normal and 30,449 attack samples; only 940 normal samples are misclassified as attacks and 2,486 attacks are missed.

The class-wise metrics computed from this confusion matrix are given in Table 4. Model-2 reaches an overall accuracy of 93.35%, with



Figure 10 Training and validation accuracy versus epochs, demonstrating the progressive improvement and generalization performance of the Model-2 across epochs.

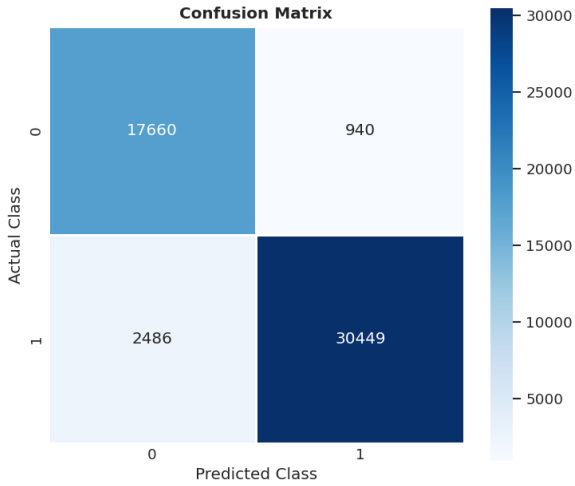


Figure 11 Confusion matrix for binary classification, where Class 0 denotes normal traffic and Class 1 denotes intrusion attacks, for Model-2.

F1-scores of 0.91 for normal traffic and 0.95 for intrusion attacks. Compared with Model-1, the recall for attacks increases from 0.86 to 0.92 and the precision for normal traffic from 0.79 to 0.88.

Table 4 Class-wise performance comparison of Model-2 across different classes.

Class	Prec.	Recall	F1	Acc. (%)
Class 0 (Normal)	0.88	0.95	0.91	93.35
Class 1 (Attack)	0.97	0.92	0.95	

The ROC curve of Model-2 (Figure 12) gives an area under the curve (AUC) of 0.966, indicating that the two classes are well separated across decision thresholds.

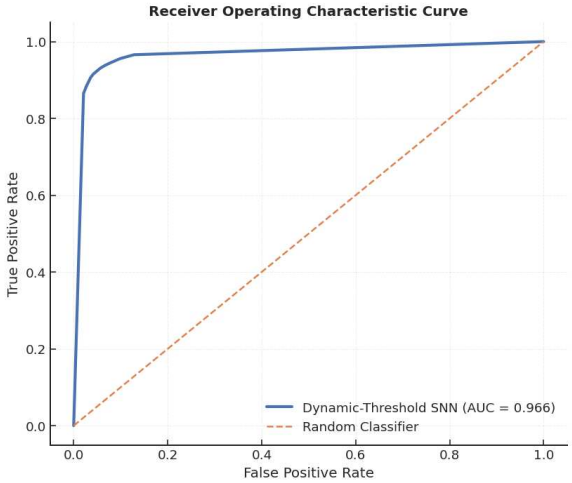


Figure 12 ROC curve for Model-2 showing the true positive rate vs false positive rate plot.

Figure 13 shows the membrane potential and the dynamic threshold of a hidden neuron of Model-2 over the five time steps (0–4). For this neuron, the membrane potential stays at approximately 0.0, while the dynamic threshold rises from approximately 1.37 to 1.55. The membrane potential never approaches the threshold, so no spikes are emitted during this window.

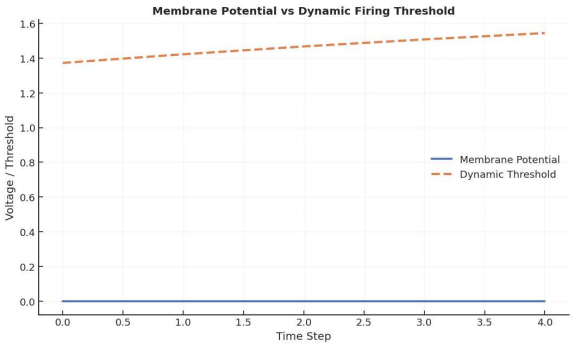


Figure 13 Illustration of the membrane potential vs dynamic firing threshold for Model-2.

As Figure 14 shows, the threshold stays above the base value of 1.0 and well inside the clamped bounds [0.5, 2.0]; the clipping in Eq. (17) prevents runaway drift.

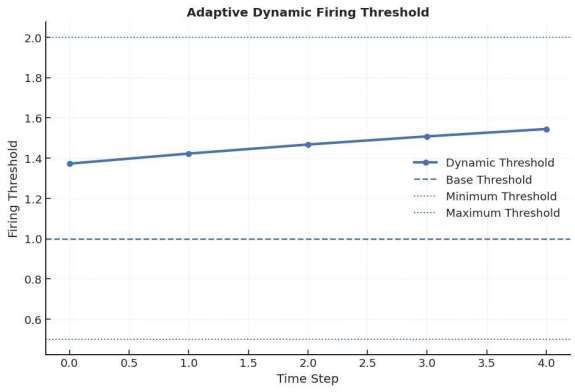


Figure 14 Variation of the adaptive dynamic firing threshold in LIF neurons for Model-2.

The spike raster of the 64 neurons of the first hidden layer (Figure 15) shows that most neurons are either persistently active or persistently silent over the 5 time steps, while a smaller subset toggles in an alternating pattern, indicating sparse, largely stable population coding with some temporal variation.

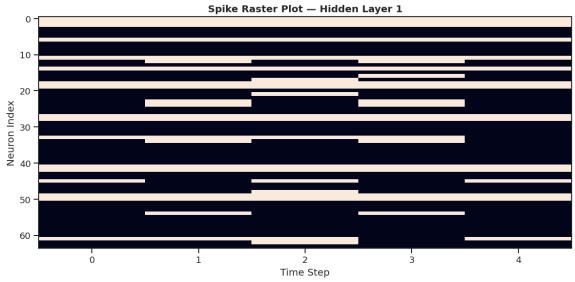


Figure 15 Visual description of the spike raster plot for hidden layer 1 used in Model-2.

6.4 Comparison of Model-1 and Model-2

Table 5 compares the two models, where FPR is the false-positive rate (normal traffic flagged as an attack) and FNR the false-negative rate (missed attacks). Model-2 improves the accuracy by 5.24 percentage points and the macro F1-score by 5.41 points over Model-1. It also lowers the false-positive rate from 8.37% to 5.05% and the false-negative rate from 13.87% to 7.55%, so nearly half of the attacks missed by Model-1 are detected by Model-2. The Matthews correlation coefficient (MCC), which is robust to class imbalance, increases from 0.757 to 0.860.

Table 5 Comparison of Model-1 and Model-2 on the test set. Precision, recall and F1-score are macro-averaged.

Metric	Model-1	Model-2	Change
Accuracy (%)	88.11	93.35	+5.24
Precision (%)	86.82	92.33	+5.51
Recall (%)	88.88	93.70	+4.82
F1-score (%)	87.51	92.92	+5.41
MCC	0.757	0.860	+0.103
FPR (%)	8.37	5.05	−3.32
FNR (%)	13.87	7.55	−6.32
AUC-ROC	–	0.966	–

6.5 Performance Comparison with State-of-the-art

It is observed from Table 6 that several state-of-the-art approaches report higher overall accuracy, particularly deep hybrid models such as SNN + CNN (98.80%) [8], CNN-SNN with blockchain (99.15%) [13], and the task-scaled SNN (98.65%) [23]. Higher accuracy is often achieved through increased architectural complexity, deeper feature extractors, or system-level integrations, and these results are obtained on different datasets and experimental settings, so they are not directly comparable. However, when we take a closer look at these models, we find that many high-accuracy models rely on convolutional, recurrent, or attention layers and do not explicitly address feature overlap or enforce class-aware representation learning. In contrast, the proposed Model-2 achieves an overall accuracy of 93.35% with a compact fully connected 39:64:32:2 spiking network, with high recall for normal traffic (0.95) and high precision for attack detection (0.97), relying on a representation-centric design (PCA, DAE, and SNN) rather than complex hybrid layers. Therefore, while some existing methods achieve higher raw accuracy, the proposed model offers a compact, interpretable, and computationally

Table 6 Performance comparison with state-of-the-art methods. The proposed models are evaluated on NSL-KDD (binary classification) with macro-averaged precision, recall and F1-score; the other results are taken from the respective studies and datasets.

Ref.	Model	Acc.	Prec.	Recall	F1
Wang *et al.* [8]	SNN+CNN	98.80	95.00	97.60	96.00
Xie and Chen [9]	Conv-SNN	96.85	97.31	97.88	96.72
Karthik *et al.* [10]	Transformer–Spiking hybrid	96.50	94.00	93.00	93.00
Mustafa *et al.* [12]	RNN–SNN (HRSNN)	98.72	98.55	98.52	98.54
Dinler [23]	Task-scaled SNN	98.65	–	96.00	96.50
Živadinović and Simić [24]	Resource-efficient SNN	96.33	–	–	95.70
Proposed Model-1	Rate-encoded LIF-SNN	88.11	86.82	88.88	87.51
Proposed Model-2	DC-injected adaptive LIF-SNN	93.35	92.33	93.70	92.92

efficient solution suited to resource-constrained deployment.

7 Conclusion and Future Work

In this paper, we propose an IDS that integrates statistical feature reduction, class-aware discriminative representation learning, and SNN-based classification to address the challenges of class imbalance, feature overlap, and computational efficiency in modern network systems. The proposed class-aware DAE, trained with class-weighted objectives and an optimization strategy, learns compact and separable latent representations and mitigates representation collapse for minority attack classes. Building on these structured representations, two SNN-based models are developed: Model-1 uses rate encoding with a fixed firing threshold, and Model-2 uses direct current injection into the membrane of LIF neurons together with a bounded adaptive dynamic threshold. Experimental results on the NSL-KDD dataset show that Model-2 improves the accuracy from 88.11% to 93.35% and the macro F1-score from 87.51% to 92.92% over Model-1, while reducing false alarms from 8.37% to 5.05% and missed attacks from 13.87% to 7.55%. The sparse activation and event-driven nature of SNNs make the model well-suited for deployment in edge and neuromorphic computing environments.

The present study has some limitations. The SNN models are evaluated for binary classification on a single dataset, and the energy efficiency is analyzed using an operation-count model rather than measured on neuromorphic hardware. In addition, the individual contributions of direct current injection, the adaptive threshold, and the change of loss function to the improvement of Model-2 have not yet been isolated through an ablation study.

In future work, the model will be extended to multi-class detection and validated on further datasets such as UNSW-NB15 [4]. Online and incremental learning will be explored for evolving attacks. Since SNNs are themselves vulnerable to spike-domain adversarial perturbations [33], backdoors [34], and hardware trojans [35], the robustness of the proposed models against such attacks will also be studied. Finally, deployment on neuromorphic hardware will provide practical validation of energy efficiency and latency.

Declarations

Funding. No funding was received for conducting this study.

References

[1] NCC Group: Annual Cyber Threat Intelligence 2025. Global ransomware attacks increased by 50% year-on-year, reaching 7,874 incidents in 2025 (2026). https://www.nccgroup.com/newsroom/ncc-group-annual-cyber-threat-intelligence-2025/

[2] Cybersecurity Ventures: Official Cybercrime Report 2025. Cybercrime costs are projected at $10.5 trillion annually in 2025 and $12.2 trillion annually by 2031 (2025). https://cybersecurityventures.com/official-cybercrime-report-2025/

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

[4] Moustafa, N., Slay, J.: Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6 (2015). IEEE

[5] Maass, W.: Networks of spiking neurons: the third generation of neural network models. Neural Networks **10**(9), 1659–1671 (1997)

[6] Maciag, P.S., Kryszkiewicz, M., Bembenik, R., Lobo, J.L., Ser, J.D.: Unsupervised anomaly detection in stream data with online evolving spiking neural networks. Neural Netw. **139**, 118–139 (2021) https://doi.org/10.1016/j.neunet.2021.02.017

[7] Davies, M., *et al.*: Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro **38**(1), 82–99 (2018) https://doi.org/10.1109/MM.2018.112130359

[8] Wang, Z., Ghaleb, F.A., Zainal, A., Siraj, M.M., Lu, X.: An efficient intrusion detection model based on convolutional spiking neural network. Sci. Rep. **14**(1), 7054 (2024) https://doi.org/10.1038/s41598-024-57691-x

[9] Xie, Y., Chen, H.: A novel method for effective intrusion detection based on convolutional spiking neural networks. J. King Saud Univ. Comput. Inf. Sci. **36**(2), 101975 (2024) https://doi.org/10.1016/j.jksuci.2024.101975

[10] Karthik, M.G., Keerthika, V., Mantena, S.V., Siri, D., Yeluri, L.P., Lella, K.K., Ganesh, B.R.: Energy-efficient intrusion detection with a protocol-aware transformer–spiking hybrid model. Scientific Reports **16**, 7095 (2026) https://doi.org/10.1038/s41598-026-37367-4

[11] Zarzoor, A.R., Al-Jamali, N.A.S., Qader, D.A.A.: Intrusion detection method for internet of things based on spiking neural network and decision tree method. Int. J. Electr. Comput. Eng. **13**(2), 2278–2288 (2023) https://doi.org/10.11591/ijece.v13i2.pp2278-2288

[12] Mustafa, M., Babiker, S.M.E., Mustafa, Y.E.A.: Hybrid recurrent with spiking neural network model for enhanced anomaly prediction in iot networks security. Frontiers in Artificial Intelligence **8** (2025) https://doi.org/10.3389/frai.2025.1651516

[13] Muthusenthil, B., Devi, K.: Deep convolutional spiking neural network and blockchain-based intrusion detection framework for enhancing privacy and security in cloud computing environment. Comput. Stand. Interfaces **97**, 104126 (2026) https://doi.org/10.1016/j.csi.2025.104126

[14] S N, P., D, S., Shelke, N., Pimpalkar, A., Jang Bahadur Saini, D.K., Kumar, G.H.: Event-driven intrusion detection systems using spiking neural networks for edge and iot security. In: 2025 5th International Conference on Soft Computing for Security Applications (ICSCSA), pp. 41–47 (2025). https://doi.org/10.1109/ICSCSA66339.2025.11171294

[15] Gretsch, R., Beyeler, M., Lau, J., Sherwood, T.: Single spike artificial neural networks. In: Proc. 52nd Annu. Int. Symp. Comput. Archit. (ISCA), pp. 916–929 (2025). https://doi.org/10.1145/3695053.3731027

[16] Rehan, R., Sahran, S., Sani, N.S., Alyasseri, Z.: A hybrid abc-based online evolving spiking neural network for unsupervised anomaly detection in streaming data. PeerJ Comput. Sci. **11**, 3184 (2025) https://doi.org/10.7717/peerj-cs.3184

[17] Mia, M.Z.A., Bal, M., Lu, S., Nishibuchi,

G.M., Chelian, S., Vasan, S., Sengupta, A.: Neuromorphic cybersecurity with semi-supervised lifelong learning. In: Proc. Int. Conf. Neuromorphic Syst. (ICONS), pp. 235–238 (2025). https://doi.org/10.1109/ICONS69015.2025.00043

[18] Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., Shi, L.: Direct training for spiking neural networks: Faster, larger, better. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 1311–1318 (2019)

[19] Kim, Y., Park, H., Moitra, A., Bhattacharjee, A., Venkatesha, Y., Panda, P.: Rate coding or direct coding: Which one is better for accurate, robust, and energy-efficient spiking neural networks? In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 71–75 (2022)

[20] Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., Maass, W.: Long short-term memory and learning-to-learn in networks of spiking neurons. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 31, pp. 787–797 (2018)

[21] Yin, B., Corradi, F., Bohté, S.M.: Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. Nature Machine Intelligence **3**, 905–913 (2021)

[22] Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., Tian, Y.: Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 2661–2671 (2021)

[23] Dinler, Ö.B.: Task-scaled spiking neural networks with multi-stage feature selection for resource-constrained iot intrusion detection. Cluster Computing **29**, 198 (2026) https://doi.org/10.1007/s10586-026-06075-z

[24] Živadinović, M., Simić, D.: Resource efficient internet-of-things intrusion detection with spiking neural networks. In: Proc. 19th Conf. Comput. Sci. Intell. Syst. (FedCSIS). ACSIS, vol. 40, pp. 73–78 (2024). https://doi.org/10.15439/2024F8800

[25] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)

[26] Neftci, E.O., *et al.*: Surrogate gradient learning in spiking neural networks. IEEE Signal Processing Magazine **36**(6), 61–63 (2019)

[27] Vazquez, R.A., Cachón, A.: Integrate and fire neurons and their application in pattern recognition. In: 2010 7th International Conference on Electrical Engineering Computing Science and Automatic Control, pp. 424–428 (2010). IEEE

[28] Gerstner, W., Kistler, W.: Spiking Neuron Models. Cambridge University Press, ??? (2002)

[29] Bohte, S.M., Kok, J.N., La Poutre, H.: Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing **48**(1–4), 17–37 (2002)

[30] Zenke, F., Ganguli, S.: Superspike: Supervised learning in multilayer spiking neural networks. Neural Computation **30**(6), 1514–1541 (2018)

[31] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)

[32] Horowitz, M.: 1.1 computing's energy problem (and what we can do about it). In: 2014 IEEE International Solid-state Circuits Conference Digest of Technical Papers (ISSCC), pp. 10–14 (2014). IEEE

[33] Xu, N., Mahmood, K., Fang, H., Rathbun, E., Ding, C., Wen, W.: Attacking the spike: On the security of spiking neural networks to adversarial examples. Neurocomputing **656**, 131506 (2025) https://doi.org/10.1016/j.neucom.2025.131506

[34] Li, J., Wu, B., Xia, X., Liu, X., Yi, X., Zhang, X.: Unsupervised backdoor detection and mitigation for spiking neural networks. In: Proc. 28th Int. Symp. Res. Attacks Intrusions Defenses (RAID), pp. 50–64 (2025). https://doi.org/10.1109/RAID67961.2025.00025

[35] Raptis, S., Kling, P., Kaskampas, I., Alouani, I., Stratigopoulos, H.-G.: Input-triggered hardware trojan attack on spiking neural networks. arXiv preprint arXiv:2503.21793 (2025) https://doi.org/10.48550/arXiv.2503.21793