Application of SNN and CNN Models for Intrusion Detection in Resource-Constrained Networks

Alina Fesu, Nathan Shone, Áine MacDermott, Bo Zhou, Max Eiza
School of Computer Science and Mathematics
Liverpool John Moores University, Liverpool, UK
a.fesu@2020.ljmu.ac.uk, {n.shone, a.m.macdermott, b.zhou, m.hashemeiza}@ljmu.ac.uk

***Abstract*—Deep learning models like Convolutional Neural Networks (CNNs) are widely used in Intrusion Detection Systems (IDSs), but their high energy demands and complexity limit deployment in resource-constrained environments. This paper presents a feasibility study of a lightweight yet deep Spiking Neural Network (SNN) based on Leaky Integrate-and-Fire (LIF) dynamics for sustainable IDS applications. Evaluated on the NSL-KDD dataset, the proposed SNN achieved comparable overall accuracy to a CNN, trained ~3x faster (~2.14 s/epoch), and consumed up to 5x less energy. Despite a slightly lower macro F1 score (0.58 vs. 0.62), it outperformed the CNN on rare attacks (e.g., U2R F1: 0.82 vs. 0.42) and exhibited lower test loss, indicating better calibration. Local deployment estimates showed ~65% lower CO2 emissions than cloud execution, challenging assumptions that offloading is inherently greener. Its low energy footprint, compact architecture, and fast inference latency (~0.014 ms/sample) make it well-suited for real-time traffic first - line analysis in edge or IoT environments. These findings highlight SNNs as viable, sustainable alternatives for IDS and underscore the importance of evaluating carbon emissions, not just energy use, when designing AI systems for cyber security.**

Keywords—spiking neural networks, convolutional neural networks, intrusion detection, energy efficiency, CO2 emissions, artificial intelligence, edge computing, IoT security, hybrid encoding, rate coding, carbon footprint, resource-constrained environments.

I. Introduction

Intrusion Detection Systems (IDSs) have significantly benefited from advancements in Artificial Intelligence (AI), particularly through the use of Deep Learning (DL) models. Among these, Convolutional Neural Networks (CNNs) have become increasingly popular in IDS development [1]. Their ability to automatically extract relevant features from raw data, handle complex datasets, reduce false positives, and accurately classify normal and malicious behaviour [2], [3] has made them highly effective for intrusion detection tasks.

CNNs' impressive performance comes at a cost; they are computationally intensive and require substantial hardware resources for training and inference [4], [2]. Since IDS models often process large volumes of network traffic, inspecting every packet or connection to detect potential threats, this continuous analysis can be computationally expensive and energy intensive, especially when powered by DL architectures such as CNNs. This limits their real-time responsiveness [3] and scalability, while raising concerns about energy consumption and environmental impact. Efforts to reduce CNN power consumption [5] and optimise data handling [6] have achieved only incremental improvements, failing to address the fundamental computational and energy-intensive nature of CNNs. Research suggests offloading computationally intensive models to cloud platforms in low-carbon regions [7]. While this promises scalability and emissions reductions, significant limitations remain. Global deep learning energy usage surged 300,000-fold from 2012 to 2018 [8], doubling every few months [4], and AI data center demands continue to rise exponentially [9]. The Paris AI Convention (Feb 2025) concluded that rapid expansion of AI data centers could overwhelm renewable energy infrastructure, exacerbating power shortages and emissions. Additionally, lack of transparency in data center energy sourcing complicates accurate emissions accounting; for example, Google Cloud promotes low-carbon infrastructure and carbon offsets [10], yet workload locations are abstracted [11] and offset claims rely on unverifiable assumptions. These concerns highlight the importance of developing energy-efficient IDS models as an element of responsible AI design.

Spiking Neural Networks (SNNs) offer a promising alternative by processing data only when triggered by specific events, thereby minimising unnecessary computations and energy use. Although SNNs can surpass CNNs in energy efficiency, they remain underexplored in IDS applications, and their environmental impact is not fully understood. Motivated by SNNs' potential to balance performance and efficiency, this study investigates whether they can match or exceed CNNs in attack classification while reducing computational demands and carbon emissions. To test if cloud offloading reduces emissions, we compare the carbon footprint of CNNs and SNNs during training and inference on both a personal device and a cloud platform, offering practical insight into sustainable IDS deployment in varying environments.

The main contributions of this work are as follows:

- Proposing a lightweight deep LIF-based SNN for intrusion detection, demonstrating that comparable classification performance and energy efficiency are achievable without complex or resource-heavy architectures.
- Conducting a comparative evaluation of SNN and CNN models across local and cloud platforms, assessing classification performance, and estimating energy consumption, and CO_2 emissions to inform sustainable IDS deployment strategies.
- Implementation of a hybrid encoding approach that combines rate coding for numerical values and one-hot encoding for categorical values, enabling the SNN to

effectively process mixed data types and capture relevant patterns in network data.

The remainder of this paper is structured as follows: Section II reviews related work on CNN and SNN-based IDS approaches. Section III outlines their theoretical foundations. Section IV describes the methodology. Section V presents the results, and Section VI concludes with key findings and future directions of this work.

II. Related works

The related work section provides a comprehensive overview of CNN and SNN-based IDS approaches, highlighting key innovations and limitations that justify the objectives of this study.

CNN-based Approaches

CNNs have demonstrated strong IDS performance, particularly for common attacks. A CNN in [12] achieved 99.55% detection accuracy, 99.63% detection rate (DR), and 0.12% false alarm rate (FAR) on CIC-IDS2017. A hybrid CNN with Gated Recurrent Unit (GRU) and feature fusion (CNN-GRU-FF) [13] further reduced FAR to 0.10%, with DRs of 98.22% on UNSW-NB15 and 99.68% on NSL-KDD. Compared to Recurrent Neural Networks (RNNs), CNNs achieved higher accuracy: 99% on KDD and 91.5% on CSE-CIC-IDS2018, versus 93% and 65% by RNNs, particularly excelling at Denial-of-Service (DoS) detection [14]. A five-layer CNN reached the lowest inference latency (2.52 ms) with high accuracy (98.81%), while CNN–Long Short-Term Memory (CNN-LSTM) models achieved slightly higher accuracy (98.84%) but with increased latency (up to 7.52 ms), illustrating a performance-speed trade-off [15]. On imbalanced data, [16] reported fast test times (1.42 s binary, 2.96 s multiclass on 22,543 NSL-KDD samples) using CNN with SMOTE (Synthetic Minority Over-sampling Technique), Focal Loss, and gradient coordination. Furthermore, CNNs with channel attention mechanisms [17] outperformed auto-encoders and hybrid approaches.

Efforts to adapt CNNs for resource-constrained environments, such as IoT and IIoT, have improved efficiency but still face limitations. CNN-LSTM-GRU [18] achieved 100% accuracy and 0% false positives (FPR) on Edge-IIoTset and 99.75% accuracy with 0.20% FPR on NSL-KDD, with latency <0.30 ms/sample. However, its complexity makes it more suited to fog computing than fully edge-based deployments. Lightweight CNNs (e.g., mCNN [19]) improve multiclass detection with ~51 s average classification time but need further tuning for imbalance. Robust Multi-Cascaded CNNs (RMC-CNN) [20] show energy-efficient performance using Multi-scale Grasshopper Optimization, though results apply to entire frameworks, limiting comparability. For energy efficiency, [21] proposed a CNN-based hybrid CPU + FPGA architecture delivering up to 4.5x higher MFLOPS/W than CPU-only, but with 85 W (CPU) and 81.45 W (CPU+FPGA) power draw—potentially limiting broader deployment on standard edge devices. [22] proposed a hybrid CNN–LSTM IDS for fog computing, achieving 99.94% accuracy (KDD-99), 92.76% multi-class accuracy (CICIoT2023), and < 0.38% FAR. Deployed on a Raspberry Pi, it reached 6.12 W power usage and 0.30 ms/sample latency, though its architectural complexity may obstruct full edge deployment.

SNN-based Approaches

Unlike CNNs, which rely on continuous-valued activations and dense matrix multiplications, SNNs process information through discrete spikes, enabling event-driven computation and improved energy efficiency [5]. The computation in a LIF neuron is relatively simple: it integrates the input over time and, when the threshold is reached, it "fires" and resets. This is less computationally demanding than convolution operations, matrix multiplications, and activations in CNNs. However, SNNs with Leaky Integrate-and-Fire (LIF) neurons, though computationally simple, present training challenges due to their non-linear dynamics [23]. Non-leaky variants have shown improved trainability and even outperform DNNs under certain conditions [23]. To improve IDS performance, IDS-SNNDT [24] combines a Non-Leaky Integrate-and-Fire SNN with a decision tree, achieving low inference latency (1.96 ms) and energy use (4.91 μJ), outperforming IDS-DNN (6.74 ms, 9.61 μJ) and IDS-SNNTLF (3.20 ms, 6.52 μJ), though CNN-based models still surpass it on certain attacks. Other hybrid SNN approaches, such as HESADM [25] and BAbSNS [26], integrated SNNs with traditional ML techniques or bio-inspired optimisation to improve detection accuracy, particularly for unknown attacks. While promising, these models often increase complexity or depart from lightweight designs.

Environmental Impact of IDS

While some IDS studies highlight power efficiency, they are often limited to measuring electrical energy usage [27] or execution time [21], without explicitly addressing the broader environmental impact, such as the carbon footprint associated with energy consumption. Carbon footprint refers to the total greenhouse gas (GHG) emissions generated by human activity, with carbon dioxide (CO_2) being a major contributor to climate change [28], [7]. A direct link exists between energy consumption and CO_2 emissions, as computational infrastructure often relies on unsustainable power grids. This emphasises the importance of evaluating IDS models not only for performance but also for sustainability. While [28] measured the carbon emissions of CNNs in a non-cyber security context, our work is the first, to our knowledge, to assess CO_2 emissions for CNNs applied to IDS. Additionally, we extend this analysis by introducing SNNs to investigate their energy efficiency and sustainability. This study provides a comparative analysis of CNNs and SNNs, offering insights into their performance and environmental impact across cloud and local computing .

III. Background & Rationale of study

CNNs [29], originally designed for grid-like data, are widely used in computer vision tasks [30] such as image classification, object detection, and segmentation. In intrusion detection, CNNs analyse network traffic data similarly, extracting hierarchical features through convolutional layers, followed by activation functions like Rectified Linear Unit (ReLU) for non-linearity [31]. Pooling layers (e.g., max-pooling) reduce feature dimensions to enhance computational efficiency and mitigate overfitting [32], while dropout layers deactivate neurons during

training to further improve generalisation [33]. During training, CNNs employ supervised learning mechanisms, where input data passes through convolutional, pooling, and dense layers. Learnable filters and activation functions transform data into feature maps, with final dense layers handling classification or regression tasks [12]. The output is compared to true labels using a loss function, such as cross-entropy, to quantify errors. Back-propagation, combined with gradient descent optimisation, such as Adam [33] or its variants, iteratively updates weights and biases to minimise the loss function. Through multiple iterations of forward propagation, error calculation, and weight updates, CNNs learn to extract meaningful features and achieve accurate predictions.

SNNs mimic brain activity by transmitting information via discrete spikes, using Integrate-and-Fire (IF) or LIF neurons [34]. These event-driven models offer improved energy efficiency over traditional networks. LIF neurons accumulate input until a threshold is reached, then fire and reset—introducing a refractory period that helps stabilise learning and reduce overfitting. Non-leaky IF neurons omit this decay, offering simpler dynamics for specific tasks. Real-world inputs are converted to spikes via encoding: rate encoding uses spike frequency (robust but compute-heavy), while temporal encoding encodes information in spike timing (sparse and efficient but noise-sensitive) [5]. Adam [33] variants can effectively handle the sparse gradients typical in SNNs.

These foundational concepts underpin the application of CNNs and SNNs in intrusion detection systems, forming the basis for the methodology described next.

IV. Methodology

Building on Section II, this study compares a CNN and a lightweight, energy-efficient SNN for intrusion detection, evaluating performance and environmental impact. Both models were implemented in PyTorch [35], trained on the same data, and tested under consistent local and cloud conditions. Rather than assuming cloud offloading is greener [7], we test this claim following rising concerns over AI's energy demands. The next sub-sections detail the model architectures, pre-processing, and evaluation metrics.

Dataset Analysis and Pre-processing

We use the NSL-KDD dataset [36], a standard intrusion detection benchmark with 125,973 training and 22,544 test samples, each containing 43 features (24 integers, 15 floats, 4 categorical). Forty-one attack types were grouped into five classes (Normal, DoS, R2L, Probe, U2R). The *difficulty_level* meta-label was excluded.

Both CNN and SNN pipelines used consistent pre-processing: missing numerical values were imputed with training means, and categorical features were one-hot encoded and reindexed in the test set for consistency. For CNNs, numerical features were Min-Max scaled to [0, 1]; for SNNs, we applied log-scaled rate encoding (log1p - Min-Max - firing rate) to generate continuous firing rates. We tested firing rates from 50-255 Hz. Low rates (e.g., 50 Hz) led to under activation and poor recall (Class 2 = 0.06), while higher rates improved accuracy (up to 76.01%) and rare-class detection (e.g., at 255 Hz: Class 2 = 0.15, Class 4 = 0.28). We chose 175 Hz as a balanced trade-off (75.75% accuracy; Class 2 = 0.15, Class 4 = 0.33. Compared to fixed-bin encoding, continuous rate encoding avoided sparsity and preserved input detail, improving feature representation and classification.

All pre-processed features were concatenated into final input tensors. We used stratified sampling in both CNN and SNN pipelines to ensure balanced label distributions. It improved stability and rare-class generalisation. The original training set (125,973 samples) was split into training (100,778) and validation (25,195),while the test set (22,544 samples) was held out for final evaluation.

Proposed Models

We implement a single spike LIF SNN and 1-Dimensional (1D) CNN, providing a fair comparison, since not all models reviewed use the same metrics in their evaluation. The parameters were fine-tuned through empirical experimentation on the dataset. Experiments were conducted on Google Colab [37] (T4 GPU, Linux, Python 3.10.12, 12.675 GB RAM, Intel Xeon CPU @ 2.20GHz) and a personal laptop (Windows 10, Python 3.11.5, 15.825 GB RAM, Intel i7-10870H CPU @ 2.20GHz, NVIDIA RTX 3070 GPU).

1) LIF Model: We implemented a single-spike LIF model with a threshold of 1, matching the scaled input range [0, 1]. SNNs with 0-3 hidden layers were tested using LIF-only or LIF+ReLU activations. Non-ReLU models collapsed to majority predictions. While the 3-layer ReLU SNN improved Class 2 F1 (0.31 vs 0.21), the 2-layer variant achieved better Class 4 F1 (0.41 vs 0.30), and lower runtime, and was selected as final. A time constant τ = 1.0, improved test accuracy (75.17% vs 74.90) and macro F1 (0.59 vs 0.55) compared to 1.5, with higher Class 4 F1 (0.43 vs 0.26), despite a slight drop in Class 2 F1 (0.26 vs 0.20). A leak of 0.5 balanced generalisation and rare-class recall, achieving F1 scores of 0.27 (Class 2), 0.39 (Class 4), and a macro F1 of 0.60. Lowering leak to 0.2 increased Class 4 F1 to 0.43 but reduced Class 2 F1 to 0.21. Finally, a reset (hyperpolarisation) value of -1 outperformed 0.5 in test accuracy (75.17% vs 74.83), macro F1 (0.59 vs 0.56), and rare-class F1 (Class 2: 0.26 vs 0.21; Class 4: 0.43 vs 0.34). The spiking behavior is described by the equations:

a) Membrane potential updates:

$$V(t+1) = V(t) \cdot e^{-\frac{1}{\tau}} + X(t) \tag{1}$$

where *V(t+1)* is the updated membrane potential; *V(t)* is the membrane potential at time-step *t*; *τ\tau* is the membrane time constant set to 1.5, and *X(t)* is the input at time *t*.

b) Spike generation:

$$S(t) = \begin{cases} 1, & \text{if } V(t) \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where *S(t)* represents the spike output at time step *t*, and *θ* is the membrane potential threshold set to 1 in this model.

c) Membrane potential reset:

$$V(t+1) = \begin{cases} h & \text{if } S(t) = 1 \\ V(t)(1-\lambda) & \text{otherwise} \end{cases} \quad (3)$$

where $V(t+1)$ is the membrane potential at the next time step; h (hyperpolarise) is the reset value if a spike occurs; $S(t)=1$ indicates a spike at time t; and λ - *leak* is the factor by which the membrane potential decreases if no spike occurs.

2) CNN Model: The CNN processes 1D inputs through two convolutional layers (filters: 32-64-128, kernel size: 3, padding: 1), each followed by max pooling (kernel size: 2). Compared to a deeper three-layer version, the two-layer model achieved slightly lower overall test accuracy (74.04% vs 74.40%) but significantly better rare-class detection (Class 2 F1: 0.33 vs 0.19; Class 4 F1: 0.27 vs 0.03), with faster training (~8.4s vs ~13.9s/epoch), offering a better trade-off between efficiency and rare-class performance. Two fully connected layers follow: fc1 (128 neurons) and fc2 (output). We select dropout 0.02 vs 0.05 improved test accuracy (76.48% vs 76.18%) and rare-class metrics for Class 2 (recall/F1: 0.21/0.31 vs 0.12/0.21) and Class 4 (precision/recall/F1: 0.80/0.12/0.21 vs 0.70/0.10/0.18). The mathematical operations are detailed below:

a) Convolutional layers:

$$Y_{i,j}^{(l)} = \mathrm{ReLU}\left(\sum_{k=1}^{C_{\text{in}}^{(l)}} X_{i,k}^{(l-1)} \cdot W_{k,j}^{(l)} + b_j^{(l)}\right) \quad (4)$$

where $Y_{i,j}^{(l)}$ is the output feature at position *(i,j)* in the *l*-th layer (i.e., (conv1,2); $C_{\text{in}}^{(l)}$ is the number of input channels; $X_{i,k}^{(l-1)}$ is the input from the previous layer; $W_{k,j}^{(l)}$ is the weight of the convolutional filter, and $b_j^{(l)}$ is the bias term.

b) Pooling layer:

$$Z_{i,j}^{(l)} = \max_m \left(Y_{i,m}^{(l)}\right) \quad (5)$$

where $Z_{i,j}^{(l)}$ is the output after applying the max pooling to the convolutional output $Y_{i,m}^{(l)}$; m represents the pooling region.

c) Fully connected layers:

$$z = W^{(2)}(\mathrm{Dropout}(\mathrm{ReLU}(W^{(1)} x_{\text{flat}} + b^{(1)}), p)) + b^{(2)} \quad (6)$$

where $z \in \mathbb{R}^C$ is the output score vector for each class C, x_{flat} is the flattened 1D input; $W^{(1)}x_{\text{flat}} + b^{(1)}$ is the linear transformation of fc1, followed by *ReLU* activation and *Dropout* with probability p; $W^{(2)}$ and $b^{(2)}$ are the weights and biases of fc2 producing the final scores for each class.

We trained both models using cross-entropy loss. The SNN used LIF neurons with spiking outputs and linear layers for gradient-based training. Adafactor (lr=0.007) achieved the best results - test accuracy of 76.84%, macro F1 of 0.58, and U2R F1 of 0.35 - outperforming Adam, Adagrad, and Adamax (learning rate 0.005), with minor increase in epoch time. For the CNN, Adam (lr=0.005) is the best overall optimiser, with the highest test accuracy (79.23%), strong rare class 2 detection (F1 = 0.25), and competitive execution time (~9.9s/epoch). While Adamax (0.005) slightly outperforms in class 4 detection (U2R F1 = 0.23), it sacrifices test accuracy (76.03%) and has lower class 2 recall (0.09). Adagrad and Adafactor perform worse on test accuracy and rare class detection, making them less suitable.

Batches were used to prevent out-of-memory (OOM) on the Colab platform. Batch size 256 with shuffle=True was selected for the CNN, achieving the best trade-off: 77.05% test accuracy, R2L F1 = 0.26, U2R F1 = 0.45, and ~5.9s/epoch. While batch size 128 (shuffle = False) performed similarly (76.16%, R2L F1 = 0.24, U2R F1 = 0.40), it trained slower (~6.7s/epoch). Shuffling improved generalisation. For the SNN, batch size 128 (shuffle=False) gave the highest accuracy (76.69%) and U2R F1 = 0.33, but batch 256 (shuffle = True) trained fastest (~1.9s/epoch) and achieved the best rare-class F1s (R2L = 0.22, U2R = 0.40), making it ideal for efficient deployment. During training, both models updated parameters per batch and validated performance after each epoch. The CNN used batch-wise validation and testing (batch size 128), while the SNN evaluated these sets in a single pass. Metrics recorded included loss, accuracy, precision, recall, F1-score, execution time, and energy usage via Carbontracker [38]. Post-training, both models were evaluated on the test set for performance and sustainability.

V. Results and Discussion

A. Model Training and Evaluation Performance

The performance metrics observed during both training and validation are summarised in Table 1. Over 20 epochs, both models achieved high accuracy, with the CNN slightly ahead: 99.61% training, 99.57% validation, and losses of 0.0106 (train) and 0.0150 (val). The SNN followed closely with 99.05% training, 99.12% validation, and losses of 0.0267 and 0.0300. While the CNN offered better generalisation, the SNN trained 3x faster (~2.14s/epoch vs. 6.0s), making it well-suited for efficient, low-resource deployments.

Table 1. Training, Validation Performance, and Execution Time Metrics for SNN and CNN Models Across 20 Epochs

Epoch	Training and Validation									
	Train Loss		*Train Accuracy*		*Validation Loss*		*Validation Accuracy*		*Execution Time (seconds)*	
	SNN	*CNN*	*SNN*	*CNN*	*SNN*	*CNN*	*SNN*	*CNN*	*SNN*	*CNN*
1	0.0875	0.1120	97.39	96.47	0.0376	0.0559	98.72	99.09	2.42	7.22
5	0.0318	0.0197	99.01	99.32	0.0321	0.0178	98.99	99.44	2.04	5.96
10	0.0284	0.0143	99.00	99.52	0.0310	0.0173	99.06	99.43	2.03	6.04
15	0.0270	0.0112	99.06	99.60	0.0306	0.0152	99.04	99.52	1.93	5.90
20	0.0267	0.0106	99.05	99.61	0.0300	0.0150	99.12	99.57	1.99	6.06

B. Validation and Test Performance

The classification performance of the SNN LIF and CNN models was evaluated on both the validation and test sets.

On the validation set, both models achieved excellent performance on common classes (Normal, DoS, Probe), with F1-scores above 0.98. For rare classes, the SNN outperformed the CNN significantly on U2R, achieving an F1-score of 0.82 vs. 0.42, and higher recall (0.79 vs. 0.40). CNN slightly outperformed the SNN on R2L (F1 = 0.90 vs. 0.84), but the SNN still maintained high precision (0.89). Overall, the SNN achieved a better macro-average F1 (0.93 vs. 0.86), indicating stronger rare-class generalisation, while the CNN achieved slightly higher weighted average metrics due to better performance on dominant classes (Table *2*).

Table 2. Classification Report for the Validation Set

Attack Classes/ Metric	Classification Report (Validation)					
	Precision		*Recall*		*F1 Score*	
	SNN	*CNN*	*SNN*	*CNN*	*SNN*	*CNN*
Normal	0.99	1.00	0.99	1.00	0.99	0.99
DoS	0.99	1.00	1.00	1.00	1.00	1.00
R2L	0.89	0.90	0.80	0.90	0.84	0.90
Probe	0.98	0.99	0.98	0.99	0.98	0.99
U2R	0.87	0.44	0.79	0.40	0.82	0.42
Accuracy					0.99	1.00
Macro Avg	0.94	0.87	0.91	0.86	0.93	0.86
Weighted Avg	0.99	1.00	0.99	1.00	0.99	1.00

On the test set, both models achieved similar accuracy (Table 3). The CNN achieved slightly higher accuracy (77% vs. 75%), macro F1 (0.62 vs. 0.58), and weighted F1 (0.73 vs. 0.72), while also performing better on Probe and U2R (F1 = 0.70 and 0.45 vs. 0.65 and 0.40, respectively). However, the SNN outperformed the CNN on Normal (F1 = 0.81 vs. 0.79) and had a lower test loss (1.88 vs. 2.96), indicating better overall calibration. Both models struggled with R2L, achieving low recall (< 0.15), though precision remained high. While CNN offers stronger macro-level metrics, the SNN remains competitive, particularly in terms of loss and Normal-class stability, reinforcing its potential for efficient deployment in constrained settings.

Table 3. Classification Report for the Test Set

Attack Classes/ Metric	Classification Report (Test)					
	Precision		*Recall*		*F1 Score*	
	SNN	*CNN*	*SNN*	*CNN*	*SNN*	*CNN*
Normal	0.70	0.67	0.97	0.96	0.81	0.79
DoS	0.92	0.94	0.77	0.79	0.83	0.86
R2L	0.91	0.92	0.14	0.15	0.22	0.26
Probe	0.60	0.82	0.72	0.61	0.65	0.70
U2R	0.78	0.66	0.27	0.34	0.40	0.45
Accuracy					0.75	0.77
Macro Avg	0.80	0.81	0.56	0.58	0.58	0.62
Weighted Avg	0.79	0.82	0.75	0.76	0.72	0.73

C. Model Robustness Assessment

ROC Curves: (Fig. 1), ROC analysis, confirms that the CNN offers stronger class separability, particularly for rare classes like R2L and U2R (AUC: 0.93 and 0.94, respectively), while the SNN performed comparably on Normal and DoS traffic. This reinforces the CNN's advantage in overall predictive power, though the SNN maintained respectable performance with lower computational costs.

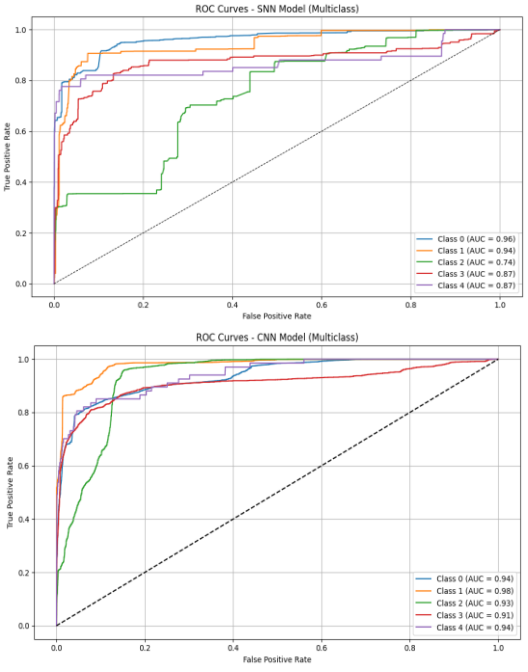

Fig. 1. ROC curves for multi-class classification on the NSL-KDD dataset. Subfigure (a) displays the ROC curves for the SNN model, and subfigure (b) shows the CNN model. Each curve corresponds to one of the classes: Normal (0), DoS (1), R2L (2), Probe (3), and U2R (4), with AUC values in the legend.

D. Carbon Footprint Tracking

The energy consumption for model training and inference was tracked per epoch using the Carbontracker library [38]. Tracking was initiated and stopped around the training loop (which includes per-epoch validation) and again separately around the final test set evaluation, allowing independent measurement of training and inference energy usage. CPU, GPU, and RAM consumption were recorded during each phase. To ensure privacy and avoid external data uploads, the Carbontracker online dashboard was not used; instead, CO_2 emissions were computed locally from the recorded energy consumption data, using a fixed carbon intensity factor of 44 gCO2/kWh based on the UK's regional average as of 18/08/2024 [39], providing a consistent and privacy-preserving estimate. This rate also served as a conservative proxy for the Colab platform, as Google cites low-carbon data center locations but does not disclose specific regions [11]. Emissions were computed using the formula:

$$gCO2 = kWh \times Carbon\ Intensity\ (gCO2/kWh) \quad (7)$$

As shown in Table 4, the SNN model demonstrated significantly greater energy efficiency and lower carbon emissions than the CNN across both platforms. On the personal laptop (estimated via constant CPU mode by Carbontracker due to the lack of real-time telemetry support on Windows), the SNN consumed approximately 0.000476 kWh (0.0209 gCO2), compared to the CNN's 0.001161 kWh (0.0511 gCO2). On Google Colab's [37] T4 GPU, the SNN used an estimated

0.001366 kWh (0.0601 gCO2), while the CNN consumed 0.006563 kWh (0.2888 gCO2). Direct comparisons with prior work were limited by missing emissions data, these results highlight the sustainability benefits of energy-efficient models like SNNs and illustrate how energy profiling can inform greener IDS development.

Table 4. Energy Consumption and CO2 Emissions Comparison for SNN and CNN Models on Google Colab T4 GPU and Personal Laptop

Energy Metric	*Colab T4 GPU*		*Personal Laptop*	
	SNN	***CNN***	***SNN***	***CNN***
RAM kWh	0.000137	0.000547	0.000070	0.000185
GPU kWh	0.000142	0.001126	0.000142	0.000272
CPU kWh	0.001229	0.004890	0.000264	0.000704
Total kWh	0.001366	0.006563	0.000476	0.001161
gCO2	0.060104	0.288772	0.020944	0.051084

Additionally, the SNN's GPU consumption on the personal laptop was just 0.000142 kWh, nearly half that of the CNN (0.000272 kWh), indicating strong potential for low-power deployment in resource-constrained environments such as IoT or edge devices.

E. Comparison with Existing Work

Unlike [21] who estimate power draw at 85 W for the CPU and 81.45 W for their CPU+FPGA setup, our study directly measures energy consumption across CPU, GPU using Carbontracker, we recorded a total of ~22.5 W CPU usage and ~32.78 W GPU usage, highlighting the practical energy efficiency of lightweight SNNs in real-world, edge-compatible deployments. We measured inference latency at 0.014 ms/sample over 1,000 test records (network connections). Compared to the five-layer CNN in [15], which achieved 98.81% accuracy with 2.52 ms/sample latency, our LIF-based SNN delivers comparable accuracy with ~14x lower latency (0.014 ms/sample). Unlike CNN–LSTM models [15], which incur higher inference costs (up to 7.52 ms), our model maintains both low latency and energy efficiency. While [16] report fast total test times (1.42s binary, 2.96s multiclass) the SNN offers ~2.14s real-time inference and low-power operation without added complexity. Against the CNN-LSTM model by [22], which reported 0.30 ms/sample latency and 6.12 W peak power during the prediction phase on a Raspberry Pi, the SNN consumed only 0.000344 kWh across training and inference combined. Averaged per epoch (~2.14 s), this corresponds to ~0.029 W, or over 200x lower average power usage. While the platforms differ, the comparison highlights strong suitability for ultra-low-power edge deployment. IDS-SNNDT [24] which reported 1.96 ms/sample, and IDS-DNN 6.74 ms/sample, while the LIF SNN achieves over 99% lower latency. Despite methodological differences, the per-sample latency definition aligns across studies, similar to the comparison with [22].

While CNN-based IDS models in prior work often achieve high accuracy, many rely on specialised hardware (e.g., GPUs or FPGAs [21]) or involve compute-heavy multi-layer designs [23]. This limits their practicality in constrained IoT or IIoT environments, where real-time response and low power consumption are critical. In contrast, the LIF-based SNN delivers competitive performance through simplicity. Finally, we compared the SNN's performance against CNN-based IDS models evaluated on NSL-KDD (Table 5). While it does not match the state-of-the-art results of deeper models like CNN-1D [23], CNN-CA [17], or CNN-GRU-FF [13], the SNN delivers competitive accuracy, precision, and recall given its simplicity. With a weighted-average F1 score of 0.72 and accuracy of 0.75, it presents a promising lightweight deep alternative. Its minimal energy footprint and architectural efficiency highlight the potential of SNNs as first-layer or edge-deployable classifiers in sustainable IDS systems, especially where environmental and computational constraints apply.

Table 5. Comparison with Existing CNN Approaches

Model	Multi-Class Classification			
	Accuracy	***Precision***	***Recall***	***F1 Score***
SNN (test set results)	0.75	0.79	0.75	0.72
CNN (test set results)	0.77	0.82	0.76	0.73
CNN-1D [23]	0.9564	0.9565	0.9564	0.9561
CNN CA [17]	0.99	0.99	0.99	0.99
CNN-GRU-FF [13]	99.86	99.68	-	99.68
N-LIF Original [23]	0.9717	0.9721	0.9717	0.9718
N-LIF Resampled [23]	0.9931	0.9931	0.9931	0.9931

VI. CONCLUSION

This study evaluated the feasibility of a lightweight SNN with LIF dynamics for energy-efficient intrusion detection. While not outperforming the CNN on some rare-class F1 scores, the SNN achieved comparable overall accuracy, trained 3x faster, and consumed up to 5x less energy during training and inference. These findings highlight the SNN's potential for efficient, low-power deployment in edge and IoT environments. Although cloud platforms are considered greener due to provider commitments to carbon neutrality, our results suggest that local execution of small models like SNNs can emit significantly less CO_2 in practice. In our case, laptop deployment, reduced emissions by approximately 65% compared to cloud. This comparison should be interpreted with caution, as the cloud estimate relies on a fixed proxy value, as cloud energy sourcing remains opaque, echoing earlier concerns about transparency. Because CPU energy was estimated in constant mode in both environments, future work will incorporate real-time power tracking for improved accuracy. Additionally, since experiments focused on NSL-KDD, testing on more modern IoT/IIoT datasets is needed to validate generalisability. Given their efficiency, SNNs are promising as first-line, real-time traffic analysers or as components of decentralised, energy-aware IDS. Future work will also explore long-term emissions in operational settings and integration into green federated learning frameworks.

REFERENCES

[1] L. Mohammadpour, T. C. Ling, C. S. Liew, and A. Aryanfar, 'A Survey of CNN-Based Network Intrusion Detection', *Applied Sciences*, vol. 12, no. 16, 2022, doi: 10.3390/app12168162.

[2] A. Redhu, P. Choudhary, K. Srinivasan, and T. K. Das, 'Deep learning-powered malware detection in cyberspace: a contemporary review', *Front Phys*, vol. 12, Mar. 2024, doi: 10.3389/fphy.2024.1349463.

[3] A. Deshmukh and K. Ravulakollu, 'An Efficient CNN-Based Intrusion Detection System for IoT: Use Case Towards

Cybersecurity', *Technologies (Basel)*, vol. 12, no. 10, p. 203, Oct. 2024, doi: 10.3390/technologies12100203.

[4] D. H. R. R. ; G. peter Borowiec, 'The environmental consequence of deep learning', Oct. 2021.

[5] K. Yamazaki, V. K. Vo-Ho, D. Bulsara, and N. Le, 'Spiking Neural Networks and Their Applications: A Review', Jul. 01, 2022, *MDPI*. doi: 10.3390/brainsci12070863.

[6] I. Goodfellow, Y. Bengio, and A. Courville, 'Deep Learning', MIT Press, 2016, ch. 9, pp. 326–366. Accessed: Jan. 01, 2024. [Online]. Available: http://www.deeplearningbook.org

[7] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, 'Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning', *J. Mach. Learn. Res.*, vol. 21, no. 1, Jan. 2020, Accessed: Dec. 15, 2023. [Online]. Available: https://jmlr.org/papers/volume21/20-312/20-312.pdf

[8] R. Parmar, 'Why we need responsible computing', Nov. 2021. Accessed: Jan. 04, 2024. [Online]. Available: https://www.bcs.org/media/8065/responsible-computing-091121.pdf

[9] International Energy Agency, 'Electricity 2024 Analysis and Forecast to 2026', 2024. [Online]. Available: Electricity 2024 Analysis and Forecast to 2026

[10] Google, 'Google's Carbon Offsets: Collaboration and Due Diligence '. Accessed: Jan. 14, 2024. [Online]. Available: https://static.googleusercontent.com/media/www.google.com/en//green/pdfs/google-carbon-offsets.pdf

[11] Google, 'Colab Enterprise locations'. Accessed: Jan. 15, 2024. [Online]. Available: https://cloud.google.com/colab/docs/locations

[12] A. H. Halbouni, T. S. Gunawan, M. Halbouni, F. A. A. Assaig, M. R. Effendi, and N. Ismail, 'CNN-IDS: Convolutional Neural Network for Network Intrusion Detection System', in *2022 8th International Conference on Wireless and Telematics (ICWT)*, 2022, pp. 1–4. doi: 10.1109/ICWT55831.2022.9935478.

[13] Y. Imrana, Y. Xiang, L. Ali, A. Noor, K. Sarpong, and M. A. Abdullah, 'CNN-GRU-FF: a double-layer feature fusion-based network intrusion detection system using convolutional neural network and gated recurrent units', *Complex & Intelligent Systems*, vol. 10, no. 3, pp. 3353–3370, Jun. 2024, doi: 10.1007/s40747-023-01313-y.

[14] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, 'CNN-Based Network Intrusion Detection against Denial-of-Service Attacks', *Electronics (Basel)*, vol. 9, no. 6, 2020, doi: 10.3390/electronics9060916.

[15] S. Ding and G. Wang, 'Research on intrusion detection technology based on deep learning', in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, 2017, pp. 1474–1478. doi: 10.1109/CompComm.2017.8322786.

[16] B. Gan, Y. Chen, Q. Dong, J. Guo, and R. Wang, 'A convolutional neural network intrusion detection method based on data imbalance', *J Supercomput*, vol. 78, no. 18, pp. 19401–19434, Dec. 2022, doi: 10.1007/s11227-022-04633-x.

[17] F. Alrayes, M. Zakariah, S. Amin, Z. Khan, and J. Alqurni, 'CNN Channel Attention Intrusion Detection System Using NSL-KDD Dataset', *Computers, Materials & Continua*, vol. 79, pp. 1–10, Jun. 2024, doi: 10.32604/cmc.2024.050586.

[18] A. Khacha, R. Saadouni, Y. Harbi, C. Gherbi, S. Harous, and Z. Aliouat, 'Robust Intrusion Detection for IoT Networks: an Integrated CNN-LSTM-GRU Approach', in *2023 International Conference on Networking and Advanced Systems (ICNAS)*, IEEE, Oct. 2023, pp. 1–6. doi: 10.1109/ICNAS59892.2023.10330519.

[19] S. A. Ajagbe, J. B. Awotunde, and H. Florez, 'Ensuring Intrusion Detection for IoT Services Through an Improved CNN', *SN Comput Sci*, vol. 5, no. 1, p. 49, 2023, doi: 10.1007/s42979-023-02448-y.

[20] K. Sakthidasan Sankaran and B.-H. Kim, 'Deep learning based energy efficient optimal RMC-CNN model for secured data transmission and anomaly detection in industrial IOT', *Sustainable Energy Technologies and Assessments*, vol. 56, p. 102983, 2023, doi: https://doi.org/10.1016/j.seta.2022.102983.

[21] L. A. Maciel, M. A. Souza, and H. C. Freitas, 'Energy-Efficient CPU+FPGA-Based CNN Architecture for Intrusion Detection Systems', *IEEE Consumer Electronics Magazine*, vol. 13, no. 4, pp. 65–72, Jul. 2024, doi: 10.1109/MCE.2023.3283730.

[22] H. Alzahrani, T. Sheltami, A. Barnawi, M. Imam, and A. Yaser, 'A Lightweight Intrusion Detection System Using Convolutional Neural Network and Long Short-Term Memory in Fog Computing', *Computers, Materials and Continua*, vol. 80, no. 3, pp. 4703–4728, 2024, doi: https://doi.org/10.32604/cmc.2024.054203.

[23] S. Zhou and X. Li, 'Spiking Neural Networks with Single-Spike Temporal-Coded Neurons for Network Intrusion Detection', Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.07803

[24] A. R. Zarzoor, N. A. Shiltagh Al-Jamali, and D. A. Abdul Qader, 'Intrusion detection method for internet of things based on the spiking neural network and decision tree method', *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 2, p. 2278, Apr. 2023, doi: 10.11591/ijece.v13i2.pp2278-2288.

[25] K. Demertzis and L. Iliadis, 'A hybrid network anomaly and intrusion detection approach based on evolving spiking neural network classification', *Communications in Computer and Information Science*, vol. 441, pp. 11–23, 2014, doi: 10.1007/978-3-319-11710-2_2.

[26] M. Eugine Prince *et al.*, 'An Optimized Deep Learning Algorithm for Cyber-Attack Detection', in *Smart Innovation, Systems and Technologies*, vol. 371, Springer, 2023, pp. 465–472. doi: 10.1007/978-981-99-6706-3_39.

[27] Y. Xie and H. Chen, 'A novel method for effective intrusion detection based on convolutional speaking neural networks', *Journal of King Saud University - Computer and Information Sciences*, vol. 36, no. 2, p. 101975, 2024, doi: https://doi.org/10.1016/j.jksuci.2024.101975.

[28] A. K. Zakaria, P. K. Mensah, A. F. Adekoya, F. Umar Bawah, and K. D. Asante, 'Measuring Carbon Emission of a Convolutional Neural Network', 2023. doi: 10.2139/ssrn.4644598.

[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[30] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, 'A review of convolutional neural networks in computer vision', *Artif Intell Rev*, vol. 57, no. 4, p. 99, Mar. 2024, doi: 10.1007/s10462-024-10721-6.

[31] V. Nair and G. E. Hinton, 'Rectified Linear Units Improve Restricted Boltzmann Machines', in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, in ICML'10. Madison, WI, USA: Omnipress, 2010, pp. 807–814. Accessed: Jan. 05, 2024. [Online]. Available: https://dl.acm.org/doi/10.5555/3104322.3104425

[32] S. Naseer *et al.*, 'Enhanced Network Anomaly Detection Based on Deep Neural Networks', *IEEE Access*, vol. 6, pp. 48231–48246, 2018, doi: 10.1109/ACCESS.2018.2863036.

[33] E. M. Dogo, O. J. Afolabi, and B. Twala, 'On the Relative Impact of Optimizers on Convolutional Neural Networks with Varying Depth and Width for Image Classification', *Applied Sciences*, vol. 12, no. 23, 2022, doi: 10.3390/app122311976.

[34] Y. Venkatesha, Y. Kim, L. Tassiulas, and P. Panda, 'Federated Learning With Spiking Neural Networks', *IEEE Transactions on Signal Processing*, vol. 69, pp. 6183–6194, 2021, doi: 10.1109/TSP.2021.3121632.

[35] PyTorch, 'PyTorch', 2024, Accessed: Apr. 08, 2024. [Online]. Available: https://pytorch.org/

[36] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, 'A detailed analysis of the KDD CUP 99 data set', in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6. doi: 10.1109/CISDA.2009.5356528.

[37] Google, 'Google Colab'. Accessed: Jan. 10, 2024. [Online]. Available: https://colab.research.google.com/#scrollTo=Wf5KrEb6vrkR

[38] L. F. W. Anthony, B. Kanding, and R. Selvan, 'Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models', Jul. 2020, Accessed: Jan. 06, 2024. [Online]. Available: https://github.com/lfwa/carbontracker/tree/master

[39] National Grid ESO, 'Carbon Intensity Dashboard'. [Online]. Available: https://www.nationalgrideso.com/future-energy/our-progress-towards-net-zero/carbon-intensity-dashboard