Efficacy of Spiking Neural Networks for Intrusion Detection Systems

Leonard Knapp
FZI Research Center for Information Technology
Karlsruhe, Germany
knapp@fzi.de

Sven Nitzsche
FZI Research Center for Information Technology
Karlsruhe, Germany
nitzsche@fzi.de

Matthias Börsig
FZI Research Center for Information Technology
Karlsruhe, Germany
boersig@fzi.de

Alexandru Vasilache
FZI Research Center for Information Technology
Karlsruhe, Germany
vasilache@fzi.de

Ingmar Baumgart
FZI Research Center for Information Technology
Karlsruhe, Germany

Juergen Becker
Karlsruhe Institute of Technology
Karlsruhe, Germany

***Abstract*—Protection against potential threats is paramount in computer networks and requires robust security measures. However, traditional rule-based Intrusion Detection Systems (IDSs) often fail to adapt to dynamic environments, prompting the exploration of innovative solutions such as Neural Network (NN)-based approaches. Previous advances have primarily focused on conventional NNs. Only more recent studies researched the use of Spiking Neural Networks (SNNs) for IDSs; however, they rely on pre- or post-processing steps in their methods, which interferes with the analysis of the actual applicability of SNNs for IDSs. This study aims to overcome this deficit by analyzing the efficacy of SNNs as the sole data processor for IDSs, i.e., without using any non-essential processing outside of the network ("bare" SNNs). Through extensive experimentation on the NSL-KDD, CIC-IDS-2017, CIC-IOT-2023, and AWID3 datasets, we examined various configurations of bare SNNs, alongside conventional NNs, and Recurrent Neural Networks (RNNs) for comparison. The results demonstrate that SNNs can achieve robust performance for IDSs without the pre- or post-processing steps required by other studies. In detail, the bare SNNs achieved higher or similar accuracy for all datasets compared to the other NN models. Furthermore, a comparative analysis reveals a competitive advantage of SNNs over the other NN models in generating fewer false positives. The results of this study suggest that SNN-based IDSs are a promising direction to strengthen network security. However, further research is essential to ensure broader applicability and scalability.**

***Index Terms*—Spiking neural networks, Artificial neural networks, Intrusion detection, Computer networks, Machine learning**

I. Introduction

Modern computer networks are essential for storing and processing large amounts of data. Ensuring the security of these networks against potential attackers is paramount. Traditionally, security measures have relied on systems such as Intrusion Detection Systems (IDSs) to detect and prevent unauthorized access. However, traditional rule-based IDSs require constant updates to keep up with evolving attack strategies, rendering them inefficient in dynamic environments such as university networks, enterprise infrastructures, or cloud systems [1], [2].

Recent advances have introduced Neural Network (NN)-based approaches that address these limitations by offering self-learning capabilities and adaptability. Previous studies have primarily focused on conventional NNs; only more recent studies have started exploring the use of Spiking Neural Networks (SNNs) for IDSs. SNNs are often referred to as the third generation of NNs [3]. In particular, they show promise in efficiently handling time-dependent data while conserving computational resources and energy. Most studies using SNNs for IDSs have relied on pre- or post-processing steps, which obscure the evaluation of actual applicability.

This study overcomes this deficit by exploring the applicability and efficacy of "bare" SNNs in the context of IDSs. "Bare" means the SNN is the sole data processor stripped of all non-essential processing outside the network. We thoroughly investigated the use of bare SNNs in IDSs by extensive comparative analysis with conventional NNs and Recurrent Neural Networks (RNNs). The comparative analysis required us to generate a data basis, as the related work does not provide suitable data for such comparisons.

We make the following contributions:

- Exploring the quality of different structures and parameters of SNNs, conventional NNs, and RNNs for the NSL-KDD [4], CIC-IDS-2017 [5], CIC-IOT-2023 [6], and AWID3 [7] datasets.

This research is funded by the German Federal Ministry of Research, Technology and Space as part of the project "SASVI", funding no. 16KIS1577.

DOI: https://doi.org/10.1109/Cyber-AI66431.2025.11233776 URL: https://ieeexplore.ieee.org/document/11233776

- Comparative analysis of the quality of SNNs, conventional NNs, and RNNs in IDSs, taking into account metrics such as accuracy, false positives, and F1-score, alongside a brief investigation of their computational efficiency.
- Proposals for improving future IDS datasets to enhance SNN performance and energy efficiency in this area.
- Publication of all code and scripts on GitHub[1] for reproducibility of results.

II. Background

SNNs differ from conventional NNs in their approach to modeling neurons and encoding data, closely mimicking biological neural networks [3]. Unlike conventional NNs, which use real-valued numbers, SNNs represent data by unary spikes, where the temporal intervals between these spikes encode the information. The temporal dynamics of SNNs makes them excel at processing time-dependent data, where each data point correlates with previous ones. However, it also makes them incompatible with non-temporal datasets without prior temporal data encoding.

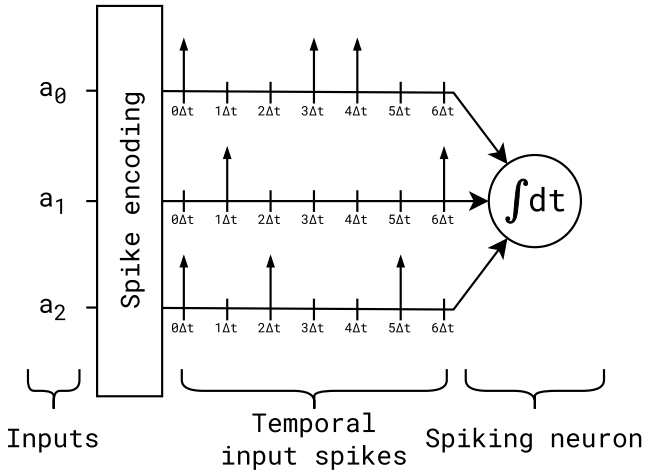

Fig. 1. Visualization of spike sequences with seven timesteps as inputs to a spiking neuron.

Spiking neurons process spike sequences (”spike trains”) using stateful activation functions, which are more complex than typical NN activations such as the Rectified Linear Unit (ReLU). These activation functions typically involve one or more differential equations describing an internal cell voltage of the spiking neuron. Furthermore, they often incorporate fine-tuning parameters, resulting in SNNs generally being more challenging to train than conventional NNs. However, while the activation functions in SNNs may be more complex, the overall computations between layers are simplified [8]. The output of a spiking neuron is binary, 0 or 1, which reduces the computations to additions instead of multiplications, i.e., a synapse weight is either added to the subsequent neuron input or not. Furthermore, with appropriate neuromorphic software/hardware implementations, neuron-level computations in SNNs can be event-driven. In an event-driven execution, neurons remain inactive for zero-valued or non-existent inputs, leading to no computational updates, which results in considerable advantages in terms of energy efficiency [9].

The implementation details vary between different models, with this work focusing on the Leaky Integrate-and-Fire (LIF) neuron model due to its low computational overhead [10]. In a LIF neuron, incoming spikes are integrated over time so that the cell voltage of the neuron increases upon receiving an input spike. Once the cell voltage exceeds a threshold, the neuron generates an outgoing spike and resets. In the absence of input spikes, the cell voltage decreases over time due to leakage currents. Consequently, a LIF neuron may not spike if it receives too few or weak inputs.

III. Related Work

Sharma and Mangat [11] first recognized the potential of using SNNs for IDSs. In their work, they presented a review discussing the application of SNNs in this area, highlighting the opportunities and challenges. Although their research provided valuable insights into the concept, it focused primarily on providing a general overview rather than presenting concrete results or a specific implementation for an IDS.

Demertzis and Iliadis [12] proposed the first implementation on IDSs using SNNs. They used evolving Spiking Neural Networks (eSNNs), which adapt their structure during learning. However, they used the KDD Cup 1999 dataset, which other studies have proven to contain significant semantic errors like duplicate entries.

Further research like from Zarzoor et al. [13] or Niu [14] introduced further approaches for the use of SNNs in IDSs. However, the majority of research we could find incorporates pre-processing algorithms or post-processing steps. Therefore, directly comparing our study with previous work is challenging since we focus on SNNs as the sole data processor.

IV. Method

For the reasons pointed out in Section I, we were required to generate our own data basis to allow a comparative analysis of conventional NNs, RNNs, and SNNs for intrusion detection. The data basis needed to be broad for a thorough investigation, which required copious amounts of networks with varying hyperparameters. The hyperparameters are listed in Table I and were determined throughout the training process by the hyperparameter optimizer Optuna [15] using a Parzen-Tree Estimator (PTE).

Different input data was essential to broaden the data basis further. As we considered conventional NNs and SNNs in this study, we required datasets compatible with both models, i.e., that contain scalar values and spike trains. However, to the best of our knowledge, no such dataset currently exists. Therefore, we decided to use existing IDSs datasets with spike encoding methods. To meet our criteria of differing input data, we selected four prominent datasets with varying features and

[1]https://github.com/nitzsche-fzi/SNN-ANN-Intrusion-Detection

TABLE I
LIST OF HYPERPARAMETERS.

Parameter	Minimum	Maximum	Options
No. hidden layers	1	3	-
No. neurons per layer	10	1000	-
Learning algorithm	-	-	SGD ADAM
Learning rate	10^{-8}	10^{-1}	-
Training steps	1	30000	-
Timesteps (SNN only)	10	300	-
Δt (SNN only)	-	-	1ms
τ_{syn}^{-1} (SNN only)	100	300	-
τ_{mem}^{-1} (SNN only)	50	150	-
$v_{threshold}$ (SNN only)	10^{-4}	1	-

labels: NSL-KDD [4], CIC-IDS-2017 [5], CIC-IOT-2023 [6], and AWID3 [7]. We acknowledge that the NSL-KDD and CIC-IDS-2017 datasets are significantly outdated but included them to enable comparison with prior research.

The extensive size of the datasets forced us to reduce them because of hard disk space limitations. Each reduced dataset was constructed by randomly selecting entries of its original dataset. Table II shows the resulting number of malicious, benign, and total entries for training and validation. Training and validation data were strictly separated.

As for the number of networks, we decided that 1000 networks of every considered NN model for every considered dataset suffices. For three different NN models and four datasets, this resulted in a total of 12000 networks that needed to be trained and evaluated. To keep the training and evaluation time feasible, we decided on the linear feed-forward fully connected network structure, as it is relatively fast to train but is also easily comparable across different NN models. To further speed up processing, we used a batch size of 64.

All networks were trained within the hyperparameter ranges of Table I and validated with exactly 10000 entries of the respective reduced validation dataset. Each entry in training and validation was randomly selected for every network to avoid the possibility of overfitting the hyperparameters to specific entries. We mapped features consisting of strings to integers and split IP addresses, MAC addresses, and similar into their respective bytes. All features were $[0; 1]$ normalized. We recognize that the AWID3 dataset contains uniquely identifying features like IP addresses. However, we decided to keep these identifying features, as a previous test run with and without these features yielded identical results.

Additionally, we decided to use binary classification to label a dataset entry as malicious or benign. A loss function that forces a binary output was required. We performed a preliminary test to evaluate the feasibility of conventional binary loss functions like Sigmoid or Binary Cross Entropy. However, we concluded that they did not converge fast enough, or their results were unclear. For this reason, we designed a modified mean absolute error function "Force Decision Loss (FDL)" that doesn't have these problems:

$$FDL(x,\hat{x}) = \begin{cases} 6 \cdot |x-\hat{x}| & \text{if } x = 0\,,\ \hat{x} < -0.15 \\ 7 \cdot |x-\hat{x}| & \text{if } x = 1\,,\ \hat{x} < -0.15 \\ 2 \cdot |x-\hat{x}| & \text{if } x = 0\,,\ \hat{x} > 0.4 \\ 2 \cdot |x-\hat{x}| & \text{if } x = 1\,,\ -0.15 \leq \hat{x} < 0.6 \\ |x-\hat{x}| & \text{otherwise} \end{cases}$$

All 12000 networks were trained and evaluated on a system consisting of two AMD EPYC 7713 CPUs, three NVIDIA RTX A6000, four NVIDIA RTX 6000 Ada GPUs, and 504 GB of RAM.

A. Training conventional NNs and RNNs

We used PyTorch [16] to model the conventional NNs and RNNs. Figure 2 depicts the network structure.

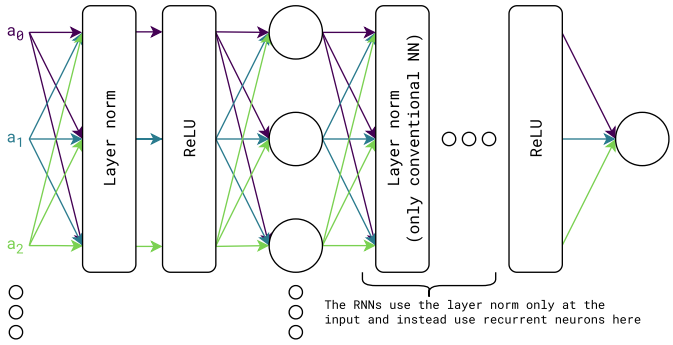

Fig. 2. Network structure of the conventional NNs and RNNs.

Optuna determined the number of hidden layers and the total number of neurons per layer according to Table I. The input and hidden layers had an identical number of neurons. The output layer was linear to a single neuron for the binary classification.

B. Training bare SNNs

The SNN framework Norse [17] was used for modeling SNNs. Figure 3 depicts the network structure.

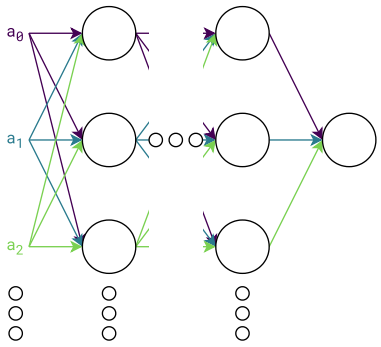

Fig. 3. Network structure of the SNNs.

The number of neurons per layer and hidden layers were determined identically as in Section IV-A. The output layer

TABLE II
QUANTITIES OF TRAINING AND EVALUATION DATA BY DATASET.

Dataset	NSL-KDD			CIC-IDS-2017			CIC-IOT-2023			AWID3		
	Malicious	Benign	Total	Malicious	Benign	Total	Malicious	Benign	Total	Malicious	Benign	Total
Train	46.5 %	53.5 %	125973	49.9 %	50.1 %	93750	50.1 %	49.9 %	126000	27.8 %	72.2 %	153764
Validate	56.9 %	43.1 %	22544	50.6 %	49.4 %	31250	49.8 %	50.2 %	42000	26.9 %	73.1 %	38750

was a single LI neuron, which is similar to a LIF neuron but does not spike. The maximum cell voltage the output LI neuron reached throughout all timesteps finally determined the output.

As mentioned, all datasets must be encoded in a temporal context to make them compatible with SNNs. Widespread spike encoding methods are rate and density encoding. We decided to use them in this study, as they require little processing time. Viewing Figure 4, their respective working principles become apparent. For rate encoding, a value close to the maximum results in a high frequency of spikes, whereas a value close to the minimum results in a single spike in the first timestep, with no further following. A spike occurs every other timestep for a value half the maximum, as shown in Figure 4a. In density encoding, all spikes are consecutive and start at the first timestep. A value close to the minimum results in no spikes at all. For a value of half the maximum, all timesteps up to half of the total number include a spike, as shown in Figure 4b.

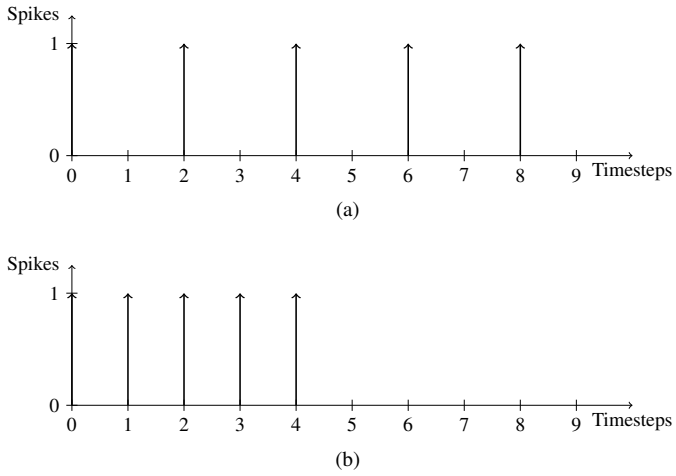

Fig. 4. Resulting spike train of a value half of the maximum encoded with (a) rate encoding / (b) density encoding.

When encoding data for SNNs, it is also important to acknowledge that input data requires time for complete network propagation due to their temporal dynamics. This fact was considered by padding the input data to fit the network depth for a given amount of timesteps.

Another important aspect of SNNs is that they have significantly more parameters than conventional NNs, as demonstrated by Table I. Generally, more parameters increase the optimization complexity and, therefore, program execution time. Furthermore, too many parameters can decrease the total quality of the optimization by the PTE, resulting in decreased accuracy of the networks. Consequently, we reduced the parameters to keep the total quality high and execution time low. Prior evaluation for the SNNs showed highly comparable results for the SGD and ADAM optimizer and networks with more than one hidden layer. Therefore, we decided to only use the ADAM optimizer for SNN training and to limit their maximum number of hidden layers to two.

V. EVALUATION

This study has created a data basis that allows comparison of different performance metrics of conventional NNs, RNNs, and SNNs in the context of intrusion detection. A total of 12000 networks have been trained and evaluated with four different datasets. Our study defined the accuracy as shown in Equation 1. The abbreviations are defined as True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

The results presented in Figure 5 show that SNNs perform better or on-par with the conventional NNs and RNNs in terms of accuracy. Accuracy improvements by SNNs in the NSL-KDD, CIC-IDS-2017, and AWID3 datasets and a similar accuracy across all NNs in the CIC-IOT-2023 dataset are observable. An evaluation of the recall (sensitivity) shows very similar behavior as indicated by Figure 6, except for the NSL-KDD dataset, in which the SNNs performed notably worse than the other NN models. However, Figure 7 reveals that while the other NN models have a higher recall for the NSL-KDD dataset, they generate far more false positives. In fact, SNNs outperformed the other NN models regarding false positives in all datasets in our study. The results of the CIC-IDS-2017 and AWID3 datasets are especially notable, in which the SNNs achieve the highest recall from all NN models with the overall lowest false positives. Everything taken into account results in the SNNs having the highest, or very similar F1-scores compared to the other NN models across all datasets, as shown by Table III. However, compared to the other NN models, SNNs required higher training and evaluation times because of their higher complexity on our non-specialized hardware, as they had to be simulated for every timestep. Our results show that simulated SNNs generally require more Floating Point Operations Per Second (FLOPS) for complete propagation, as estimated in Table IV. SNNs could overcome

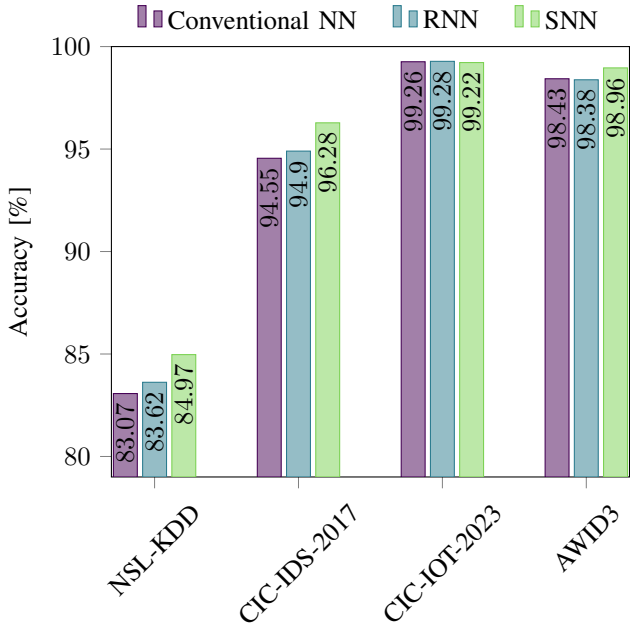

Fig. 5. Highest accuracy by NN model and dataset.

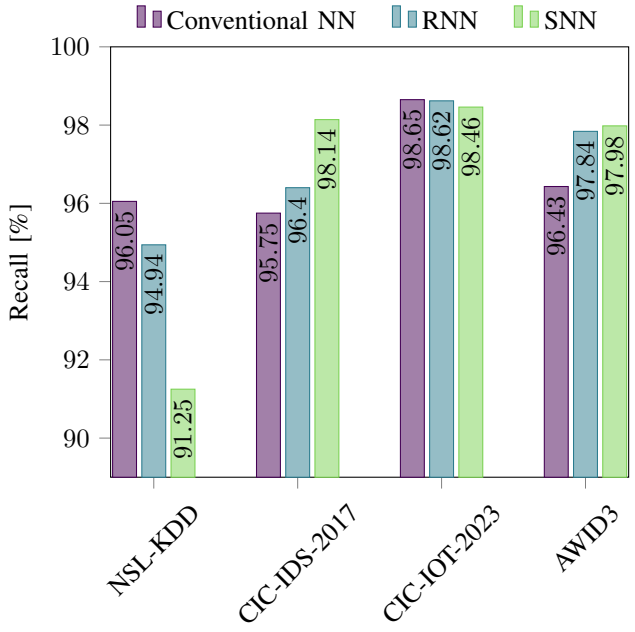

Fig. 6. Recall of the networks with respective highest accuracy by NN model and dataset.

this disadvantage by utilizing hardware with native support for sparse and event-based computation [21], [22]. We estimated the efficiency of our SNN models for native hardware by defining the number of operations the LIF neuron requires each update. Then, we multiplied this value by the number of activations across multiple real input data samples. However, as stated in Section IV, we were forced to temporally encode our scalar datasets with suitable encoding methods. For our datasets, this resulted in input spikes in 20% to 45% of all

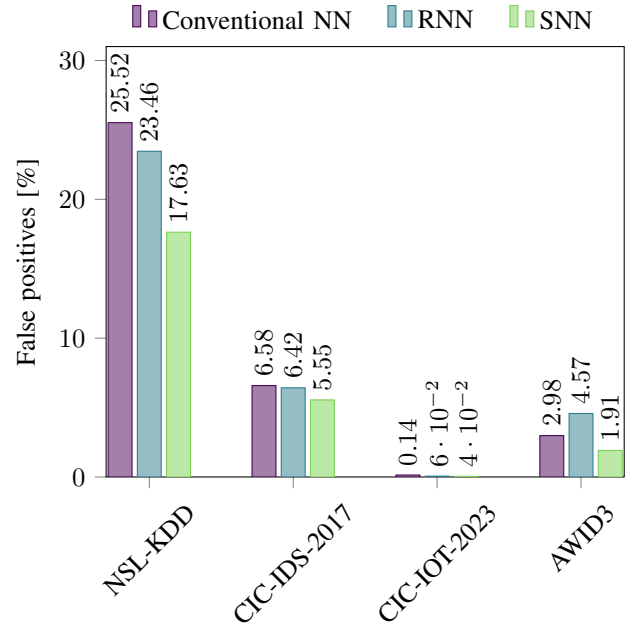

Fig. 7. False positives of the networks with respective highest accuracy by NN model and dataset.

TABLE III
F1 SCORE OF THE NETWORKS WITH RESPECTIVE HIGHEST ACCURACY BY NN MODEL AND DATASET.

Type	NSL-KDD	CIC-IDS-2017	CIC-IOT-2023	AWID3
Convent. NN	86.70 %	94.65 %	99.25 %	96.72 %
RNN	86.94 %	95.06 %	99.27 %	96.68 %
SNN	87.37 %	96.36 %	99.20 %	98.04 %

timesteps, which heavily limits sparsity and undermines the energy efficiency of SNNs. As a result, our estimated FLOPS for SNNs on native hardware is significantly higher than the FLOPS for SNNs for a sparse IDS dataset would be.

As noted in Section III, several other approaches use SNNs for IDSs. A comparison of the results can be found in Table V. Unfortunately, these publications do not provide sufficient details on important aspects such as network architecture, complexity, or size. Accuracy alone is not a very informative metric, making a comprehensive comparison with their research difficult. Furthermore, all of these studies were performed on the outdated NSL-KDD dataset. While we also used NSL-KDD for comparison purposes, our primary focus is on more recent datasets, such as CIC-IoT-2023 and AWID3, where no comparable work using SNNs exists.

VI. Challenges

The results presented in Section V suggest that the input data characteristics significantly impact the challenge of training NNs for specific tasks. This problem is particularly pronounced for SNNs due to the inherent difficulties in their training process and the lack of temporal IDS datasets to their specific requirements. However, generating such datasets is

TABLE IV
REQUIRED FLOPS OF THE MOST ENERGY EFFICIENT NETWORK WITHIN 3% OF THE HIGHEST ACCURACY NETWORKS FOR EACH DATASET.

Dataset	Convent. NN	RNN	SNN (simulated)		SNN (native)	
			Additions	Multiplications	Additions	Multiplications
NSL-KDD	$1.79 \cdot 10^5$	$1.07 \cdot 10^6$	$1.15 \cdot 10^7$	$1.15 \cdot 10^7$	$5.01 \cdot 10^5$	$2.56 \cdot 10^3$
CIC-IDS-2017	$1.21 \cdot 10^3$	$1.41 \cdot 10^3$	$2.48 \cdot 10^7$	$2.48 \cdot 10^7$	$1.25 \cdot 10^6$	$2.39 \cdot 10^3$
CIC-IOT-2023	$6.70 \cdot 10^2$	$9.20 \cdot 10^3$	$3.88 \cdot 10^5$	$3.87 \cdot 10^5$	$1.84 \cdot 10^5$	$3.46 \cdot 10^2$
AWID3	$1.92 \cdot 10^5$	$1.25 \cdot 10^5$	$4.04 \cdot 10^6$	$4.04 \cdot 10^6$	$6.56 \cdot 10^5$	$8.63 \cdot 10^2$

TABLE V
COMPARISON OF DIFFERENT SNN-BASED IDS IMPLEMENTATIONS.

Metric	Accuracy	F1	Precision	Recall	Dataset
Majumder et al. [18]	79.31%	-	-	-	NSL-KDD
Zhou and Li [19]	97.17%	97.18%	97.21%	97.17%	NSL-KDD
Vadi et al. [20]	94%	-	91%	92%	NSL-KDD
This study	84.97%	87.37%	83.81%	91.25%	NSL-KDD
	96.28%	96.36%	94.65%	98.14%	CIC-IDS-2017
	99.22%	99.20%	99.96%	98.46%	CIC-IOT-2023
	98.96%	98.04%	98.09%	97.98%	AWID3

challenging, as the only trivial indicator is the time difference between network packets, which is far from enough to identify different types of attacks reliably. More research is required to determine working methods for encoding network data into native IDS datasets for SNNs.

During our experiments, we observed significant variation in the accuracy of SNNs with minor changes to the model. For example, $[0; 1]$ normalizing the input data resulted in higher accuracy. Similarly, introducing padding in the input data to facilitate complete network propagation of the last spike improved accuracy by several percentage points. These results further underscore the need for datasets explicitly designed for SNNs, taking into account their sensitivity to how data is fed into the network.

VII. FUTURE WORK

Based on the encountered challenges, we suggest the following research directions to further investigate SNNs in intrusion detection:

- Investigate the feasibility and development of event-based datasets designed explicitly for IDSs, which could provide more granular and relevant data for SNN training. Moreover, such datasets are a prerequisite to fully taking advantage of SNNs.
- Explore the temporal dependence of datasets during conversion to better capture temporal dynamics, which could improve the performance of SNNs in intrusion detection tasks.
- Design a combined dataset with scalar and spike data for a more thorough comparison of SNN and other NN models.

VIII. CONCLUSION

This study conducted a comprehensive comparative analysis of various SNN-based IDS implementations. Our evaluation examines key performance metrics, including accuracy, F1-score, recall, and false positives, across four prominent datasets: NSL-KDD, CIC-IDS-2017, CIC-IOT-2023, and AWID3.

In addition, we also considered the resource efficiency of all NNs based on the required floating point operations for a single inference. When executed natively, we showed that SNNs can get close to the resource requirements of the other NN models but still require more operations due to low sparsity resulting from the non-optimal temporal encoding methods. A straightforward solution to increased sparsity and, therefore, resource efficiency is to use suitable temporal datasets. We determined finding such datasets to be a significant challenge when working with SNNs for IDSs since all standard datasets for intrusion detection comprise scalar values.

In conclusion, our results demonstrate that SNNs achieve robust performance in intrusion detection without requiring non-essential processing steps outside the network. We mainly observed improved or comparable performance metrics of the bare SNNs on all datasets compared to conventional NNs and RNNs. Furthermore, the SNNs showed promise in generating fewer false positives while achieving similar accuracy across diverse datasets, which highlights the significant potential of SNN-based IDS implementations in strengthening network security, especially in the dynamic and evolving environments prevalent in IoT networks. Nevertheless, further research is crucial to explore the scalability, adaptability, and real-world applicability of SNN-based IDS approaches, primarily because

fluctuations in performance arise depending on the dataset and the network architecture's complexity.

REFERENCES

[1] K. B. Ahmed Patel, Mona Taghavi and J. C. Júnior, "An intrusion detection and prevention system in cloud computing: A systematic review," in *Journal of Network and Computer Applications*, (Alpharetta, GA, USA), pp. 25–41, Elsevier, 2013.

[2] O. Kampmeier, "Why AI surpasses traditional rule-based methods in bot detection." https://www.fraud0.com/resources/why-ai-surpasses-traditional-rule-based-methods-in-bot-detection/, 8 2023.

[3] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks.," *Int. J. Neural Syst.*, vol. 19, pp. 295–308, 08 2009.

[4] 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*, (Los Alamitos, CA, USA), pp. 1–6, IEEE, 2009.

[5] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *4th International Conference on Information Systems Security and Privacy (ICISSP)*, (Setúbal, Portugal), pp. 108–116, INSTICC, SciTePress, 2018.

[6] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, 2023.

[7] E. Chatzoglou, G. Kambourakis, and C. Kolias, "Empirical evaluation of attacks against ieee 802.11 enterprise networks: The awid3 dataset," *IEEE Access*, vol. 9, pp. 34188–34205, 2021.

[8] A. Vasilache, J. Krausse, K. Knobloch, and J. Becker, "Hybrid spiking neural networks for low-power intra-cortical brain-machine interfaces," 2024.

[9] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," *CoRR*, vol. abs/1812.01739, 2018.

[10] S. Nitzsche, B. Pachideh, N. Luhn, and J. Becker, "Digital hardware implementation of optimized spiking neurons," in *2021 International Conference on Neuromorphic Computing (ICNC)*, (Los Alamitos, CA, USA), pp. 126–134, IEEE, 2021.

[11] D. Sharma and V. Mangat, "A novel approach to network intrusion detection using spiking neural networks.," *IJAEST - INTERNATIONAL JOURNAL OF ADVANCED ENGINEERING SCIENCES AND TECHNOLOGIES*, vol. 2, pp. 36–42, 11 2010.

[12] K. Demertzis and L. Iliadis, "A hybrid network anomaly and intrusion detection approach based on evolving spiking neural network classification," in *E-Democracy, Security, Privacy and Trust in a Digital World* (A. B. Sideridis, Z. Kardasiadou, C. P. Yialouris, and V. Zorkadis, eds.), (Cham), pp. 11–23, Springer International Publishing, 2014.

[13] A. Zarzoor, N. Adnan, N. Al-Jamali, and D. Aldaloo, "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, pp. 2278–2288, 04 2023.

[14] X. Niu, "Intelligent detection of network intrusion based on artificial bee colony optimized spiking neural network," *Journal of Network Intelligence*, vol. 8, pp. 1240–1255, 11 2023.

[15] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, (New York, NY, USA), p. 2623–2631, Association for Computing Machinery, 07 2019.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[17] C. Pehle and J. E. Pedersen, "Norse - A deep learning library for spiking neural networks," Jan. 2021. Documentation: https://norse.ai/docs/.

[18] P. G. Debojit Majumder, Anubhav Singh and S. Phadikar, "A novel snn-ann based ids in cloud environment," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, (Los Alamitos, CA, USA), pp. 913–918, IEEE, July 2020.

[19] S. Zhou and X. Li, "Spiking neural networks with single-spike temporal-coded neurons for network intrusion detection," in *2020 25th International Conference on Pattern Recognition (ICPR)*, vol. abs/2010.07803, (Los Alamitos, CA, USA), pp. 8148–8155, IEEE, 2021.

[20] V. R. Vadi, S. Abidin, A. Khan, and M. Izhar, "Enhanced elman spike neural network fostered blockchain framework espoused intrusion detection for securing internet of things network," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 12, p. 4634, 2022.

[21] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[22] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.