논문 24-49-11-01 The Journal of Korean Institute of Communications and Information Sciences '24-11 Vol.49 No.11
https://doi.org/10.7840/kics.2024.49.11.1497

Empirical Evaluation of SNN for IoT Network Anomaly Detection

Yeon-sup Lim◆°

ABSTRACT

Internet of Things (IoT) devices flourish fast along with the rapid progress of wireless network technologies. Since IoT devices usually deal with sensitive information from nearby users, protecting them from malicious network activities is critical. Artificial neural network (ANN) based approaches are known to be effective in detecting network anomalies. However, it is hard for IoT devices to apply such approaches due to constrained resources. Spiking Neural Networks (SNN) is a new type of neural network that requires low power consumption and computational overhead, which is proper for IoT devices. In this paper, using several network intrusion datasets, we conduct extensive experiments to compare the performance of ANN and SNN for identifying network attacks. Our experiment results demonstrate that SNN yields comparable performance to ANN in terms of accuracy and outperforms ANN in detecting frequently appearing attacks while consuming less energy.

Key Words : Network Anomaly Detection, Traffic Classification, Security, Spiking Neural Networks, IoT

Ⅰ. Introduction

The Internet of Things (IoT) is the network of physical objects (i.e., things) equipped with sensors, software, and network connectivity to collect data and exchange them with other devices and systems through the Internet. Representative examples of IoT devices are connected vehicles, smart home appliances, and digital healthcare products. Since such IoT devices deal with critical or sensitive information, malicious attackers attempt to compromise the IoT devices through networks. Several approaches have been proposed to detect network anomalies such as attacks and intrusions. In particular, artificial neural networks (ANN) are a promising solution for classifying network traffic or detecting malicious behavior. However, applying ANN-based anomaly detection on IoT devices raises various concerns, e.g., ANN requires large computational overhead with high energy consumption, which is inappropriate for IoT devices with low computing powers or limited power supply.

Spiking Neural Network (SNN) is a new type of neural network that consists of spiking neurons[1]. Different from traditional ANN neurons that activate outputs according to the weighted sum of real number inputs, spiking neurons fire output spikes according to the stimulus received by the sequence of discrete input spikes. Using discrete spikes to deliver information between neurons results in less energy consumption and lower computational complexity than ANN. Therefore, SNN can become an emerging solution for applying neural networks for anomaly detection in IoT devices. In this study, we conduct experiments for network anomaly detection using ANN and SNN across various network intrusion datasets. Based on the results of the comparative experiment,

※ This work was supported by the Sungshin Women' s University Research Grant of H20210020.
◆° First and Corresponding Author : Sungshin Women's University, Department of Convergence Security Engineering ylim@sungshin.ac.kr, 정회원
논문번호 : 202404-063-A-RN, Received April 10, 2024; Revised July 17, 2024; Accepted July 22, 2024

we demonstrate the performance of an SNN-based architecture for identifying anomalies in network traffic, which is feasible for IoT devices.

The remainder of this paper is organized as follows. Section 2 provides the background context for our work. We present the experiment setup for ANN/CNN/SNN based anomaly detection and the result of the experiments in Sections 3 and 4. Related work is reviewed in Section 5, and we conclude in Section 6.

Ⅱ. Background

2.1 Spiking Neural Networks

SNN is a biologically inspired neural network composed of spiking neurons and synapses. Spiking neurons emulate the behavior of biological neurons in the brain using discrete spikes. They receive spikes from an input source or prior neurons through synapses. These continuously received spikes stimulate the neuron, causing its membrane potential to rise. When the membrane potential of a neuron becomes larger than a particular threshold, it fires a spike to the next neuron and resets its membrane potential.

The Leaky Integrate and Fire (LIF) model enables spiking neurons to emulate the characteristic of biological neurons in which the membrane potential gradually decreases. Spiking neurons integrate the weighted sum of input spikes with leakage to compute their membrane potential. The integrated potential exceeds a threshold, and the LIF neuron emits an output spike. If no more spikes arrive, the membrane potential will continue to decrease due to the leakage. Once an output spike is activated, the LIF neuron sets its membrane potential to a resting voltage to prevent continuous unnecessary activations. Figure 1 presents an example of the behavior of the membrane potential and corresponding output spikes according to the series of input spikes.

The leakage is modeled by the solution of the following ordinary differential equation[2,3]:

$$\frac{dU_m(t)}{dt} = \frac{-[U_m(t) - U_{rest}] + R \times I(t)}{\tau}, \tag{1}$$

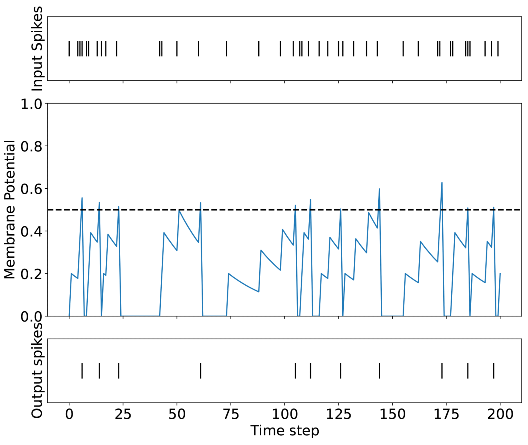


Fig. 1. Example Behavior of LIF Neuron

where $U_m(t)$ is the membrane potential at time t, U_{rest} the resting voltage of the membrane, R the resistance of the membrane, $I(t)$ the input current at time t, and τ the time constant of the neuron, respectively. Assuming the resting voltage U_{rest} is zero, Equation 1 can be rewritten as follows:

$$U_m(t+\Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) U_m(t) + \frac{\Delta t}{\tau} R \times I(t) \tag{2}$$

If there is no input current (i.e., $I(t) = 0$),

$$U_m(t+\Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) U_m(t) \tag{3}$$

Then, the decay rate of the membrane potential β is defined as:

$$\beta = 1 - \frac{\Delta t}{\tau} \tag{4}$$

Unlike ANN which continuously updates the training status per input, SNN conducts computations per infrequent spike events in information transmission. Therefore, it can operate with lower computational overhead and power consumption than ANN[4,5], making it appropriate for energy-constrained IoT devices. However, while ANN directly uses information from real number data, SNN utilizes data after converting real value into discrete spikes, which may result in performance degradation by the loss of delivered information. Note that simulating realistic neural models for SNNs on traditional computing devices is expensive in terms of computational cost[6]. Therefore, we compare the energy consumption of ANN and SNN using estimation based on the number of expected operations in Section 4.3. Neuromorphic architectures such as IBM TrueNorth[7] can practically realize SNNs with low resource overhead by directly implementing spiking neurons and synapses using electrical hardware components.

2.2 Statistical Traffic Classification

Traffic classification is one of the essential network management functions, which reveals what kinds of applications are present in a network traffic flow. It can also be used for anomaly detection, determining whether a flow is involved with malicious attacks, such as infiltration and denial of service.

A network traffic flow is a sequence of packets from one source to another destination, typically represented by a tuple of four attributes (source/destination IP addresses, source/destination ports). We can compute statistical characteristics for one flow, such as flow duration, number of packets, packet sizes, and packet inter-arrival time. Statistical traffic classification approaches utilize these computed characteristics as features for machine learning model training and its inference. Even though the existing approaches[8-16] yield feasible classification performance, they have constraints to apply to IoT devices: that is, they are based on computation-expensive machine learning algorithms inappropriate for IoT devices.

Ⅲ. Experiment Setup

3.1 Datasets

We evaluate the hierarchical anomaly detection using six widely used network intrusion datasets: KDD'99, NSL-KDD, UNSW-NB15, CICIDS-2017, CICIDS-2018, and UNR-IDD[17-22].

KDD'99[17] is a dataset that consists of network traffic instances characterized by 41 features, which can be classified as one of four attack categories: Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), and Probing attacks. Even though it is outdated, it is widely used for evaluating approaches for detecting network intrusions.

NSL-KDD[18] is a refined KDD'99 dataset. It solves inherent problems of KDD'99 such as a large number of duplicated instances and uneven levels of instance difficulty for classification by providing refined and preprocessed training and testing datasets.

UNSW-NB15[19,20] is a synthetic dataset comprised of normal and contemporary attacks activities created by IXIA PerfectStorm. We utilize the preprocessed flow-level traces with 49 features used in [19].

CICIDS 2017 and 2018 datasets[21] offer raw packet capture traces of network traffic that consist of benign and malicious network activities. We use the dataset that consists of 61 network flow statistics extracted and cleaned from the raw packet traces[23].

UNR-IDD[22] is an intrusion detection dataset of network statistics on 32 metrics observed at switch/router ports. By providing the port statistics instead of flow level ones, it enables a fine-grained network flow analysis and classification at the port level.

We apply additional data cleaning procedures on these raw datasets, such as removing unnecessary features like IP addresses, instances with null features, and classes with a few instances. Except for the datasets with predefined training and testing sets, such as NSL-KDD and UNSW-NB15, we split the entire dataset into 70% training and 30% testing data per class. We exclude classes with fewer instances than ten while generating the training and testing sets. Tables 1 and 2 summarize the number of features/labels considered and training/testing instances in the datasets and the proportion of attack instances in the test datasets (the pro-portions are the same in the training dataset).

Table 1. Dataset Summary

Dataset	Number of Considered Features	Number of Considered Labels	Number of Instances (Training, Testing)
KDD'99	36	5 (17 classes)	3428873, 1469525
NSL-KDD	37	5 (27 classes)	125940, 22508
UNSW-NB15	41	10	1778032, 762015
CICIDS-2017	68	15	1748202, 749239
CICIDS-2018	68	15	8160264, 3497268
UNR-IDD	29	6	26186, 11225

Table 2. Benign and Attack Instances in Test Datasets

Dataset	Number of Benign Instances	Number of Attack Instances	Proportion of Attack Instances (%)
KDD'99	291,835	1,177,690	80.14
NSL-KDD	9,711	12,797	56.86
UNSW-NB15	665,902	96,113	12.61
CICIDS-2017	621,547	127,692	17.04
CICIDS-2018	3,095,308	401,960	11.49
UNR-IDD	1,132	10,093	89.92

3.2 Anomaly Detection Architecture

We employ a general hierarchical architecture for detecting anomalies in network traffic, which consists of binary and multi-class classifiers. Each classifier can be built utilizing any machine learning algorithm such as traditional ANN and SNN.

The binary classifier determines whether a data instance is associated with normal or attack (intrusion) activities. To this end, the binary classifier is trained with a dataset that only includes two classes: benign for normal activities and attack for intrusion ones. The multi-class classifier identifies the attack type in an instance, assuming it is associated with attacks. The multi-class classifier is built upon a dataset with attack instances with assigned classes. Figure 2 depicts the overall network architecture to conduct the hierarchical anomaly classification.

We implement the hierarchical classifiers based on ANN and SNN using PyTorch[24] and snnTorch[25]. Considering the constrained resources in IoT devices, we employ a simple network architecture with a small

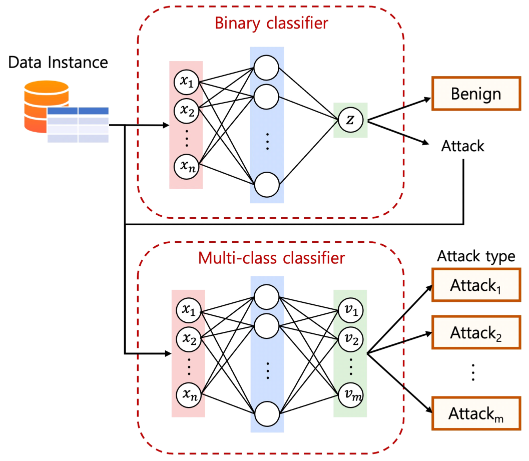


Fig. 2. Anomaly Detection Architecture

number of neurons and layers as possible. The implemented ANN and SNN classifiers have the same network architecture with input, hidden, and output layers, as shown in Figure 2. We put 100 neurons into the hidden layer and make the output layers have the number of neurons equal to that of labels in the datasets, e.g., a binary classifier has two neurons in its output layer, and KDD'99's four attack labels result in four neurons at the output layer of the multi-class classifier. The classes for data instances are determined by the output neuron with the maximum softmax in ANN and the maximum spike count in SNN. For building the ANN networks, we utilize the linear layer of PyTorch (torch.nn.linear) to fully connect each layers and the rectified linear unit function (torch.nn.ReLU) as an activation function. In the case of the SNN networks, we use the LIF layer of snntorch (snntorch.leaky) to emulate LIF neurons that integrate the weighted inputs over time and fire a spike if a condition satisfies. We set the decay rate of the membrane potential of LIF neuron to 0.95 (i.e., $\beta = 0.95$ in Equation 4). Then, we use torch.nn.linear to fully connect the LIF neurons with assigned weights. For example, the following Python code creates a layer containing five LIF neurons with $\beta = 0.95$, which accepts ten inputs and makes three outputs.

```
fc1 = torch.nn.Linear(10, 5)
lif1 = snntorch.Leaky(beta=0.95)
fc2 = nn.Linear(5, 3)
out = fc2( lif1( fc1(x) ) )
```

For the loss functions, we utilize the cross-entropy loss in PyTorch and the cross-entropy spike rate loss functions in snnTorch for ANN and SNN models, respectively. The Adam optimizer with a learning rate of 0.001 and running average parameters of (0.9, 0.999) is used for both models for the training procedure. We train both SNN and ANN models for ten epochs to reflect the constrained environments where it is not possible to train a model for a large number of epochs. For the training datasets, binary classifiers use entire datasets labeled with benign and attack classes, and multi-class classifiers use datasets in which benign instances are excluded.

To examine whether the simple ANN and SNN are able to yield comparable performance to a deep and complex network architecture, we also test a hierarchical classifier using a convolutional neural network that consists of three convolutional layers and two fully connected layers (CNN). The first convolutional layer has one input and 32 output channels, the second one has 64 output channels, and the third one has 128 output channels using ReLU as an activation function. The kernel size and stride of all convolutional layers are set to 2 and 1, respectively. Each convolutional layer applies max-pooling of which kernel size and stride are 2. The first fully connected layer makes 512 outputs using ReLU and dropout with a probability of 0.5. The second fully connected layer works the same as the output layer of ANN and SNN.

3.3 Feature Selection

The set of features selected for a machine learning model affects the performance of the resulting model[26,27]. Feature selection is a process to find the best set of features for optimizing models. It also aims to reduce the number of input features to decrease the computational cost of modeling. Several feature selection techniques are based on a statistical test of data characteristics such as information gain, correlation coefficient, and Chi-squared test.

Except for CNN, we vary the number of features used for ANN and SNN models to investigate their impact on the performance of the classifiers. To this end, we compute Chi-square scores between each feature and target class and then choose the set of features that yield the best Chi-square scores. In the case of CNN, we utilize 25 features for all the datasets.

3.4 Performance Metrics

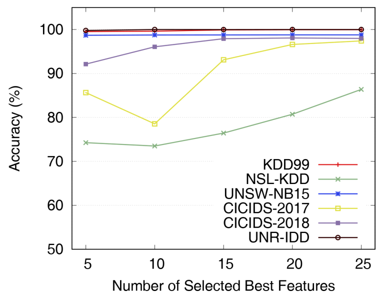
We use four metrics to evaluate the performance of anomaly detection: overall accuracy, precision, recall, and F1-score:

- Overall accuracy is the ratio of correctly classified instances to all instances in a given dataset. While this is a representative metric to demonstrate the performance of the classifier on the entire trace set, the following metrics are to evaluate the quality of classification results for each class.
- Precision is the ratio of True Positives over the sum of True Positives and False Positives, where True Positives is the number of correctly classified instances into a given class and False Positives the number of instances falsely selected to the class.
- Recall is the ratio of True Positives over the sum of True Positives and False Negatives, where False Negatives is the number of instances from a given class that are falsely labeled as another one.
- F1-score considers both precision and recall in a single metric by taking their harmonic mean.

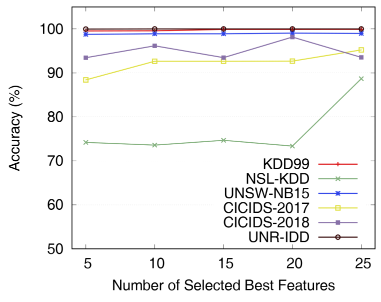
Ⅳ. Results

4.1 Binary Classification

We examine the performance of ANN and SNN binary classifiers. Figure 3 presents the overall accuracy according to the number of features for constructing models. As the number of features used for models increases, both binary classifiers yield an accuracy



(a) ANN



(b) SNN

Fig. 3. Accuracy (Binary Classification)

close to 100% except for the NSL-KDD dataset; in the case of NSL-KDD, the overall accuracy is less than 90%, even with 25 features.

Recall that attacks and benign instances in the datasets are imbalanced, as shown in Table 2. It can distort the performance of the binary classifiers. To investigate the behavior of the binary classifiers further, we inspect the precision and recall for attacks of the classifiers with the best accuracy, e.g., the binary classifier for UNR-IDD with 25 features.

Tables 3-5 list the metric values of the binary classifiers for each dataset. As shown in the tables, all binary classifiers precisely identify attacks in KDD'99, NSL-KDD, and UNR-IDD datasets, i.e., if the classifiers say one instance is an attack, it is almost always correct. Regarding precision for the other datasets, it is notable that SNN yields better values than ANN; for example, SNN obtains 0.956 and 0.980, while ANN and CNN do 0.913 and 0.875 for UNSW-NB15 and CICIDS-2017, respectively. However, for these cases, the ANN and CNN classifiers outperform the SNN one in terms of recall; the SNN classifier is less likely to find attacks than the others.

Table 3. Performance Metrics of ANN Binary Classifiers

Dataset	Accuracy (%)	Precision	Recall
KDD'99	99.96	1.000	0.999
NSL-KDD	86.37	0.972	0.783
UNSW-NB15	98.79	0.913	1.000
CICIDS-2017	97.41	0.875	0.989
CICIDS-2018	98.08	0.923	0.909
UNR-IDD	100.0	1.000	1.000

Table 4. Performance Metrics of CNN Binary Classifiers

Dataset	Accuracy (%)	Precision	Recall
KDD'99	99.96	1.000	0.999
NSL-KDD	89.11	0.971	0.832
UNSW-NB15	98.78	0.912	1.000
CICIDS-2017	97.60	0.884	0.989
CICIDS-2018	98.56	0.962	0.911
UNR-IDD	100.0	1.000	1.000

Table 5. Performance Metrics of SNN Binary Classifiers

Dataset	Accuracy (%)	Precision	Recall
KDD'99	99.91	1.000	0.999
NSL-KDD	88.68	0.972	0.825
UNSW-NB15	99.05	0.956	0.969
CICIDS-2017	95.23	0.980	0.735
CICIDS-2018	98.16	0.957	0.880
UNR-IDD	100.0	1.000	1.000

Figure 4 compares the F1-score of the binary classifiers for each dataset. For most datasets, ANN, CNN, and SNN classifiers behave similarly. All classifiers yield similar patterns in terms of F1-score; for example, they work well for KDD'99 and UNR-IDD but achieve relatively lower performance for the other datasets, such as NSL-KDD and UNSW-NB15. We observe that all the F1 scores are larger than 0.8, which means that the binary classifier will sufficiently work as a first- stage processor to filter out benign instances. Even though CNN utilizes a more complex network architecture than the others, it does not yield significant performance gains for simple binary classification. In the next section, We will examine whether CNN provides a notable enhancement in more complex classifications.

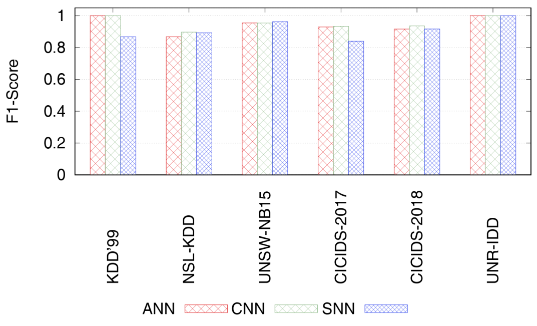


Fig. 4. F1-Score (Binary Classification)

4.2 Hierarchical Classification

We explore the performance of the multi-class classifier, which is a second-stage processor, to determine which attack is present in a traffic instance. Table 6 presents the overall accuracy of multi-class classifiers for each dataset. We set the number of features for multi-class classifiers with that from the best binary

Table 6. Accuracy (Multi-Class Classifiers)

Dataset	Accuracy (%)		
	ANN	CNN	SNN
KDD’99	99.90	99.95	99.23
NSL-KDD	74.38	71.76	73.37
UNSW-NB15	80.50	84.3	83.72
CICIDS-2017	97.35	98.15	91.55
CICIDS-2018	98.31	98.00	97.67
UNR-IDD	69.42	70.35	61.32

classifier as in Tables 3-5. As shown in Table 6, all multi-class classifiers accurately identify attack types in KDD’99, CICIDS-2017, and CICIDS-2018; their accuracy is close to 100%. However, they yield an accuracy of 60-84% for NSL-KDD, UNSW-NB15, and UNR-IDD.

Table 7 shows the overall accuracy of hierarchical classifiers in each dataset. The hierarchical classifiers yield a similar or higher accuracy than the multi-class classifier. This is because the proportion of benign instances correctly identified by the binary classifier increases the accuracy. The accuracy improvement of the hierarchical classifier is small when the proportion of benign and attack instances is even (NSL-KDD) or the proportion of attacks is higher (UNR-IDD). In the case of UNSW-NB15, where benign instances are dominant, the hierarchical classifiers obtain an accuracy higher than 96%, which is $\geq$12%p larger than that of the multiclassifier.

As with binary classification results, CNN does not notably outperform ANN or SNN although it is based on a complex architecture with large computational overheads, which results in more energy consumption. Since the intrusion datasets have insufficient input features for convolutional layers that usually process

Table 7. Accuracy (Hierarchical Classifiers)

Dataset	Accuracy (%)		
	ANN	CNN	SNN
KDD’99	99.53	99.92	99.26
NSL-KDD	78.27	77.50	79.92
UNSW-NB15	96.34	96.81	97.09
CICIDS-2017	97.06	97.40	94.84
CICIDS-2018	97.93	98.51	93.54
UNR-IDD	72.58	73.38	65.21

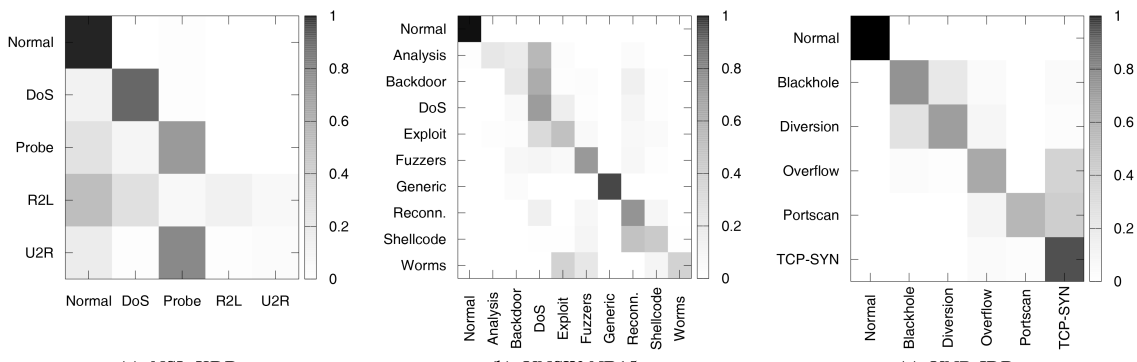
images with many pixels, the classifiers might not benefit from them.

We investigate the further detail of the hierarchical classifiers for NSL-KDD, UNSW-NB15, and UNR-IDD as representatives of even benign-attack, benign-dominant, and attack-dominant datasets. Figures 5 and 7 present the ratio of the classes into which each class instances fall. The figures are grey-scale heat maps; the darker the area, the more instances the classifier puts into the corresponding class. The x-axis is the classified class, and the y-axis is the original.

Figures 5(a), 6(a) and 7(a) exhibit that all classifiers behave similarly in the case of NSL-KDD, where the numbers of benign and attack instances are similar. Note that an imbalance between attack classes can still exist even though the ratio of benign instances is even to attack instances. U2R takes a proportion of only 4% in NSL-KDD, while DoS, R2L, and U2R do 68%, 21%, and 16%, respectively. SNN focuses more on dominant classes than ANN and CNN; for example, SNN identifies DoS instances in NSL-KDD more, while it finds fewer U2R ones than ANN and CNN. We observe this SNN behavior more explicitly in the UNSW-NB15 dataset.

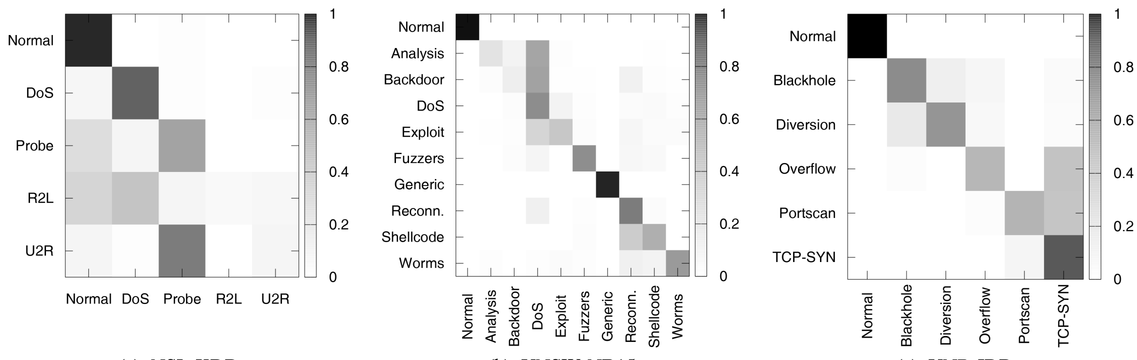
As shown in Figures 5(b), 6(b), and 7(b), SNN does not identify Analysis, Backdoor, Shellcode, and Worms in UNSW-NB15 at all, while it finds more Normal, Generic, and Exploits than ANN and CNN. Even though SNN does not work for these attacks, it yields higher overall accuracy than ANN and CNN in Table 7. This is because these attacks take insignificant portions of the dataset; thus, they do not significantly affect the overall accuracy.

To examine how well the classifier identifies all instances of a particular attack class and its impact on the overall performance, we investigate the proportion of the classes and the recall per class of ANN, CNN, and SNN in Table 8. SNN outperforms ANN and CNN in identifying the attack classes with a large number of instances, such as generic and exploits. The proportions of these attacks are 8.47% and 1.71% in the entire dataset. Note that these numbers become 67.19% and 13.55%, respectively, if only attack instances are counted. As with the result of NSL-KDD,



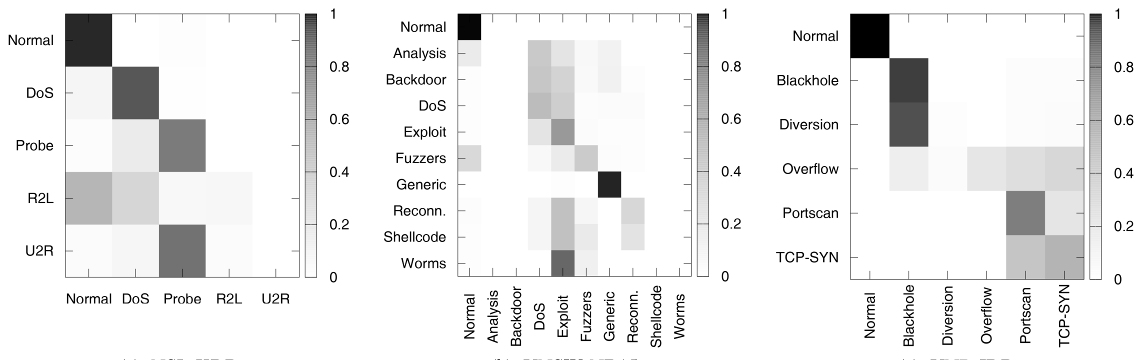
(a) NSL-KDD (b) UNSW-NB15 (c) UNR-IDD

Fig. 5. Ratio of Classified Classes (ANN)



(a) NSL-KDD (b) UNSW-NB15 (c) UNR-IDD

Fig. 6. Ratio of Classified Classes (CNN)



(a) NSL-KDD (b) UNSW-NB15 (c) UNR-IDD

Fig. 7. Ratio of Classified Classes (SNN)

SNN focuses on the dominant attack classes and yields better performance for identifying them than ANN and CNN, resulting in better overall accuracy. From the perspective of SNN as a biologically inspired neural network, it is natural for SNN to remember frequently appearing attacks well and to forget intermittent ones. If a specific attack often appears, spikes will frequently pass through neurons on the path to a corresponding output neuron, allowing them to preserve some membrane potential (i.e., excitatory and sensitive to that attack). In contrast, the neurons related to a rare attack will receive spikes intermittently, which makes it hard to sufficiently stimulate neurons to emit spikes for a corresponding output.

In the case of the UNR-IDD dataset, we again observe the same behavior of SNN focusing on classes

Table 8. Recall per Class (Hierarchical Classifier for UNSW-NB15 Dataset, sorted by proportion in dataset)

Class	Proportion (%)	ANN	CNN	SNN
Normal	87.39	0.986	0.986	0.994
Generic	8.47	0.925	0.970	0.970
Exploits	1.71	0.462	0.429	0.673
Fuzzers	0.96	0.674	0.722	0.400
DoS	0.65	0.660	0.723	0.495
Reconnaissance	0.55	0.693	0.791	0.321
Analysis	0.11	0.204	0.237	0.000
Backdoor	0.09	0.195	0.143	0.000
Shellcode	0.06	0.410	0.564	0.000
Worms	0.01	0.352	0.667	0.000

with a large proportion in Figures 5(c), 6(c), and 7(c); SNN yields lower recall than ANN for Overflow, of which proportion is only 2.76% while achieving larger recall for PortScan and Blackhole with proportions of 25.59% and 22.68%. However, SNN fails to identify Diversion attacks even though its proportion is not negligibly small (15.13%); it misclassifies almost all of them into Blackhole attacks (Blackhole is an attack to make routers/switches discard incoming packets, and Diversion is one that forces routers/switches to reroute packets away from destinations[22]). In contrast, Figures 5(c) and 6(c) show ANN and CNN are also confused when discriminating between Blackhole and Diversion, but it is less than SNN.

Table 9. Recall per Class (Hierarchical Classifier for UNR-IDD Dataset, sorted by proportion in dataset except for Normal)

Class	Proportion (%)	ANN	CNN	SNN
Normal	10.17	1.000	1.000	1.000
PortScan	25.59	0.525	0.538	0.783
TCP-SYN	23.67	0.913	0.888	0.545
Blackhole	22.68	0.698	0.730	0.939
Diversion	15.13	0.655	0.689	0.015
Overflow	2.76	0.596	0.521	0.215

4.3 Energy Consumption

In this section, we compare the energy consumption for inferencing using the classifiers based on ANN and SNN. Since it is straightforward that CNN consumes more energy and its performance is similar to ANN, we exclude CNN from this comparison. To estimate the energy consumption of the classifiers, we utilize the energy model based on the number of synapse operations.

Most of the synapse operations in ANN are multiplication and accumulation (MAC) operations between fully connected layers. Let $N_I^{(i)}$ denote the number of inputs at the *i*th ANN layer with $N_n^{(i)}$ neurons of which number of outputs is $N_O^{(i)}$. Then, the number of the synapse operations of the *i*th layer $N_{ANN}^{(i)}$ can be computed as $N_{ANN}^{(i)} = N_I^{(i)} \times N_n^{(i)} + N_n^{(i)} \times N_O^{(i)} = (N_I^{(i)} + N_O^{(i)})N_n^{(i)}$. In contrast, accumulation (AC) operations dominate in SNN since the spikes are binary indicators for the weight accumulating at the next neurons; thus, accumulations occur by spike activity so that we can estimate the number of AC operations of each SNN layer as a function of its average output spike rate $\gamma^{(i)}$. The average output spike rate at the *i*th layer $\gamma^{(i)}$ is defined as the ratio of total output spike counts to the total number of neurons at the layer. Then, the estimated number of AC operations at the *i*th layer $N_{SNN}^{(i)}$ is $N_I^{(i)} \times \gamma^{(i-1)} \times N_n^{(i)} + N_n^{(i)} \times N_O^{(i)} \times \gamma^{(i)}$. Assuming the energy consumption of MAC and AC operations are E_{MAC} and E_{AC}, we approximate the energy consumption of ANN and SNN inferences as follows[28]:

$$E_{ANN} = \left(\sum_{i=1}^{L} N_{ANN}^{(i)} \right) \times E_{MAC} \quad (5)$$

$$E_{SNN} = \left(\sum_{i=1}^{L} N_{SNN}^{(i)} \right) \times E_{AC}, \quad (6)$$

where L is the number of layers in the networks.

Table 10 presents the average output spike rate of the SNN binary and multi-class classifiers when they run one inference in the test datasets.

For E_{MAC} and E_{AC}, we assume that the considered IoT device is a sort of 45nm CMOS device running at 0.9V[29]; i.e., its E_{MAC} and E_{AC} are 3.2 pJ and 0.1 pJ, respectively. Table 11 shows the estimated energy consumption of the ANN and SNN based hierarchical

Table 10. Average Spike Rate of SNN Binary and Multi-class Classifiers for Processing One Instance in Test Datasets

Dataset	Binary		Multi-class	
	Input→ Hidden	Hidden→ Output	Input→ Hidden	Hidden→ Output
KDD'99	9.73	10.00	11.03	5.00
NSL-KDD	4.94	9.96	6.03	5.08
UNSW-NB15	4.93	10.05	8.00	2.96
CICIDS-2017	6.16	9.46	7.44	1.59
CICIDS-2018	9.18	10.0	7.81	1.43
UNR-IDD	4.20	9.93	4.53	6.36

classifiers for processing one instance in test datasets, and Figure 8 presents the ratio of the energy consumption of SNN to ANN. As shown in Table 11, the estimated energy consumption per instance of SNN for all datasets is less than 5 nJ, while the minimum of ANN is larger than 20 nJ. SNN consumes an almost consistent amount of energy (3 - 4 pJ) for one instance classification except for UNR-IDD (1.27 pJ), while CNN yields significantly different amounts of energy consumption according to the number of selected features and output classes for the datasets.

Figure 8 depicts the energy consumption ratio of SNN to ANN. SNN consumes up to 97% less energy than ANN for processing data instances. Recall that the accuracy and recall performance of SNN is comparable to ANN in Table 7. Therefore, these results demonstrate that the SNN-based approach is more suitable for IoT devices with the constraint of a limited power supply than the ANN-based one.

Table 11. Estimated Energy Consumption of Hierarchical Classifier for Processing One Instance in Test Datasets

Dataset	Estimated Energy Consumption (nJ/Instance)	
	ANN	SNN
KDD'99	28.80	3.51
NSL-KDD	48.00	3.14
UNSW-NB15	70.40	3.05
CICIDS-2017	128.00	3.81
CICIDS-2018	128.00	4.64
UNR-IDD	22.40	1.27

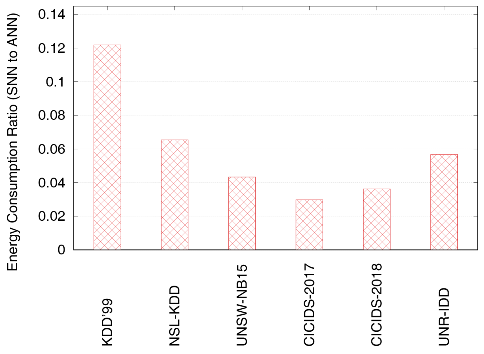


Fig. 8. Estimated Energy Consumption Ratio

V. Related Work

Several approaches have been proposed to apply machine learning (ML) algorithms to classify traffic or identify network intrusion[8-16,30,31]. However, there are few rigorous studies on utilizing spiking neural networks for intrusion detection for IoT devices with constrained resources, such as energy and computational power.

Lim and Yoo[16] examine traditional machine learning algorithms such as decision tree, random forest, and XGBoost for identifying network intrusion detection in wireless sensor networks. Kim and Choi[16] propose a network intrusion system based on edge computing. Dong et al.[14] investigate multi-class support vector machine algorithm for network traffic classification. A group of studies[8-13,31] apply various deep learning techniques such as multi-layer perceptron, convolutional neural networks, recurrent neural networks, and autoencoder. These approaches aim to classify network traffic or identify malicious network activities by utilizing statistical traffic features or manipulated network data, e.g., traffic data is converted as images for deep learning models. Since these approaches can work even with encrypted network traffic[30], they can provide an alternative for deep packet inspection, which has privacy and inoperability issues since it investigates packets' contents to identify which applications generate particular network traffic at monitoring devices. However, since they seek higher performance without considering resource usage when classifying network traffic, it is inappropriate to apply these approaches to IoT devices directly.

Rasteh et al.[32] utilize SNN for classifying encrypted traffic such as Tor browser and VPN, but it is not validated with network intrusion datasets.

Ⅵ. Conclusions

This paper evaluates ANN and SNN-based network anomaly detection schemes across various network intrusion datasets. Our experiment results demonstrate that SNN-based approaches can perform similarly to ANN for most datasets. However, we also observe that SNN does not successfully identify rarely appearing classes as it is designed to mimic a biological neural network, which tends to forget intermittent ones. Our future work will be enhancing the SNN architecture to obtain at least a similar performance to ANN in identifying infrequent classes.

References

[1] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” *Neural Netw.*, vol. 10, no. 9, pp. 1659-1671, 1997. (https://doi.org/10.1016/S0893-6080(97)00011-7)

[2] L. Abbott, “Lapicque’s introduction of the integrate-and-fire model neuron (1907),” *Brain Res. Bulletin*, vol. 50, no. 5, pp. 303-304, 1999. (https://doi.org/10.1016/S0361-9230(99)00161-6)

[3] P. Dayan and L. Abbott, *Theoretical Neuroscience*, Cambridge, MA: MIT Press, 2001.

[4] H. Mostafa, B. U. Pedroni, S. Sheik, and G. Cauwenberghs, “Fast classification using sparsely active spiking networks,” in *Proc. IEEE ISCAS*, pp. 1-4, 2017. (https://doi.org/10.1109/ISCAS.2017.8050527)

[5] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, “Long short-term memory and learning-to-learn in networks of spiking neurons,” in *Proc. Int. Conf. NIPS*, pp. 795-805, Montréal, Canada, 2018. (https://dl.acm.org/doi/10.5555/3326943.33270 17)

[6] S. Furber, “Large-scale neuromorphic computing systems,” *J. Neural Eng.*, vol. 13, no. 5, p. 051 001, Aug. 2016. (https://doi.org/10.1088/1741-2560/13/5/051001)

[7] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, et al., “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Sci.*, vol. 345, no. 6197, pp. 668-673, 2014. (https://doi.org/10.1126/science.1254642)

[8] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, “Malware traffic classification using convolutional neural network for representation learning,” in *Proc. ICOIN, IEEE*, pp. 712-717, 2017. (https://doi.org/10.1109/ICOIN.2017.7899588)

[9] Y. Zeng, H. Gu, W. Wei, and Y. Guo, “Deep－Full－Range: A deep learning based network encrypted traffic classification and intrusion detection framework,” *IEEE Access*, vol. 7, pp. 45 182-45 190, 2019. (https://doi.org/10.1109/ACCESS.2019.2908225)

[10] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, “Deep learning approach for intelligent intrusion detection system,” *IEEE Access*, vol. 7, pp. 41 525-41 550, 2019. (https://doi.org/10.1109/ACCESS.2019.2895334)

[11] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,” *J. Inf. Secur. and Appl.*, vol. 50, p. 102 419, 2020. (https://doi.org/10.1016/j.jisa.2019.102419)

[12] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, “Building an efficient intrusion detection system based on feature selection and ensemble classifier,” *Comput. Netw.*, vol. 174, p. 107 247, 2020. (https://doi.org/10.1016/j.comnet.2020.107247)

[13] S. Gamage and J. Samarabandu, “Deep learning methods in network intrusion detection: A survey and an objective comparison,” *J. Netw. and Comput. Appl.*, vol. 169, p. 102 767, 2020.

(https:// doi.org/10.1016/j.jnca.2020.102767)

[14] S. Dong, "Multi class svm algorithm with active learning for network traffic classification," *Expert Syst. with Appl.*, vol. 176, p. 114 885, 2021. (https:// doi.org/10.1016/j.eswa.2021.114885)

[15] J.-W. Kim and M.-J. Choi, "Design and implementation of a deep learning-based intrusion detection system in edge computing," *J. KICS*, vol. 47, no. 8, pp. 1114-1127, 2022. (https://doi.org/10.7840/kics.2022.47.8.1114)

[16] S. Lim and M. Yoo, "Comparison on machince learning techniques for intrusion detection in wireless sensor network," *J. KICS*, vol. 47, no. 11, pp. 1804-1814, 2022. (https://doi.org/10.7840/kics.2022.47.11.1804)

[17] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan, "KDD Cup 1999 Data," *UCI Machine Learn. Repository*, 1999. (https://doi.org/10.24432/C51C7N)

[18] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proc. IEEE Int. Conf. CISDA*, pp. 53-58, Ottawa, Ontario, Canada, 2009. (https://doi.org/10.1109/CISDA.2009.5356528)

[19] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Military Commun. and Inf. Syst. Conf.* (*Mil-CIS*), pp. 1-6, 2015. (https://doi.org/10.1109/Mil-CIS.2015.7348942)

[20] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Security J.: A Global Perspective*, vol. 25, no. 1-3, pp. 18-31, 2016. (https://doi.org/10.1080/19393555.2015.1125974)

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. Int. Conf. Inf. Syst. Secur. and Privacy*, Jan. 2018. (https://doi.org/10.5220/0006639801080116)

[22] T. Das, O. A. Hamdan, R. M. Shukla, S. Sengupta, and E. Arslan, "UNR-IDD: Intrusion detection dataset using network port statistics," in *Proc. IEEE CCNC*, pp. 497-500, 2023. (https://doi.org/10.1109/CCNC51644.2023.10059640)

[23] M. Verkerken, "*IDS Dataset Cleaning*," [Online]. Available: https://gitlab.ilabt.imec.be/mverkerk/ids-dataset-cleaning

[24] A. Paszke, S. Gross, F. Massa, et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in NIPS*, vol. 32, pp. 8024-8035, Curran Associates, Inc., 2019. (https://dl.acm.org/doi/10.5555/3454287.3455008)

[25] J. K. Eshraghian, M. Ward, E. Neftci, et al., "Training spiking neural networks using lessons from deep learning," in *Proc. IEEE*, vol. 111, no. 9, pp. 1016-1054, 2023. (https://doi.org/10.1109/JPROC.2023.3308088)

[26] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *Proc. ACM Int. Conf. Emerging Netw. EXperiments and Technol.* (*CoNEXT*), 2008. (https://doi.org/10.1145/1544012.1544023)

[27] Y.-S. Lim, H.-C. Kim, J. Jeong, C.-K. Kim, T. T. Kwon, and Y. Choi, "Internet traffic classification demystified: On the sources of the discriminative power," in *Proc. ACM Int. Conf. Emerging Netw. EXperiments and Technol.* (*CoNEXT*), New York, NY, USA, 2010. (https://doi.org/10.1145/1921168.1921180)

[28] S. Kundu, G. Datta, M. Pedram, and P. A. Beerel, "Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression," in *Proc. IEEE/CVF WACV*, pp. 3953-3962, Jan. 2021. (https:// doi.org/ 10.1109/WACV48630.2021.00400)

[29] M. Horowitz, "Computing's energy problem

(and what we can do about it)," in *Proc. IEEE ISSCC Digest of Technical Papers*, pp. 10-14, 2014.
(https://doi.org/10.1109/ISSCC.2014.6757323)

[30] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 76-81, 2019.
(https://doi.org/10.1109/MCOM.2019.1800819)

[31] G. D. L. T. Parra, P. Rad, K.-K. R. Choo, and N. Beebe, "Detecting internet of things attacks using distributed deep learning," *J. Netw. and Comput. Appl.*, vol. 163, p. 102 662, 2020.
(https:// doi.org/10.1016/j.jnca.2020.102662)

[32] A. Rasteh, F. Delpech, C. Aguilar-Melchor, R. Zimmer, S. B. Shouraki, and T. Masquelier, "Encrypted internet traffic classification using a supervised spiking neural network," *Neurocomputing*, vol. 503, pp. 272-282, 2022.
(https://doi.org/10.1016/j.neucom.2022.06.055)

Yeon-sup Lim

Feb. 2007 : B.S. in Computer Science and Engineering, Seoul National University

Feb. 2009 : M.S. in Computer Science and Engineering, Seoul National University

Feb. 2017 : Ph.D. in Computer Science, University of Massachusetts Amherst

Feb. 2017~Feb. 2020 : Research Staff Member, IBM T. J. Watson Research Center

Mar. 2020~Current : Assistant Professor, Department of Convergence Security Engineering, Sungshin Women's University

<Research Interest> Computer Networks, Mobile Computing, Machine Learning, Cloud Computing

[ORCID:0000-0001-7647-6185]