# Spiking Neural Networks with Single-Spike Temporal-Coded Neurons for Network Intrusion

**SHIBO ZHOU**

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING,
BINGHAMTON UNIVERSITY, NY

11/24/2020

# Outline



- Background
  - Network Intrusion
  - SNN
- Introduction
- Contribution
- Network Intrusion Datasets
  - NSLKDD
  - AWID
- Neuron Models in Spiking Neural Networks
  - Analyze Spiking Neuron Response
  - SNNs Built with n-LIF Neurons
- Experiments
- Conclusions

# Background-Network Intrusion

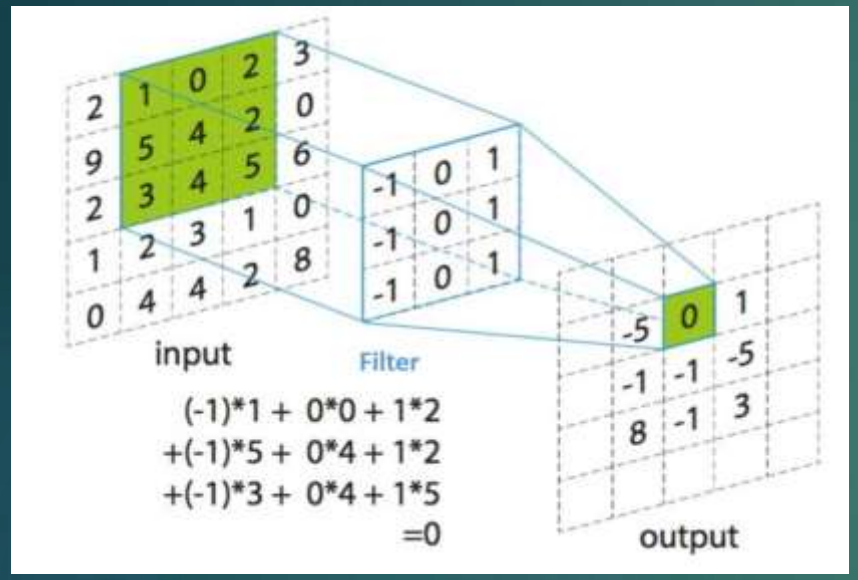


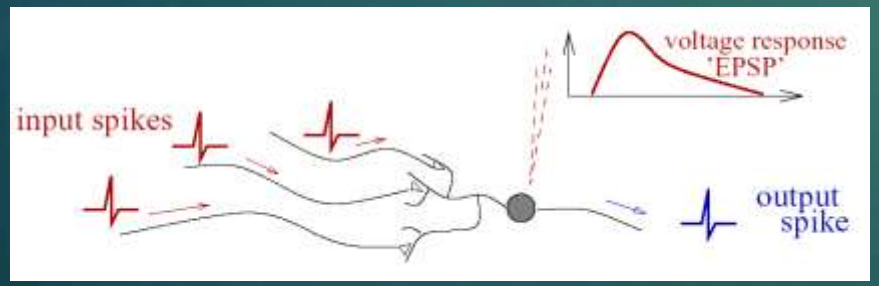



Network Intrusion

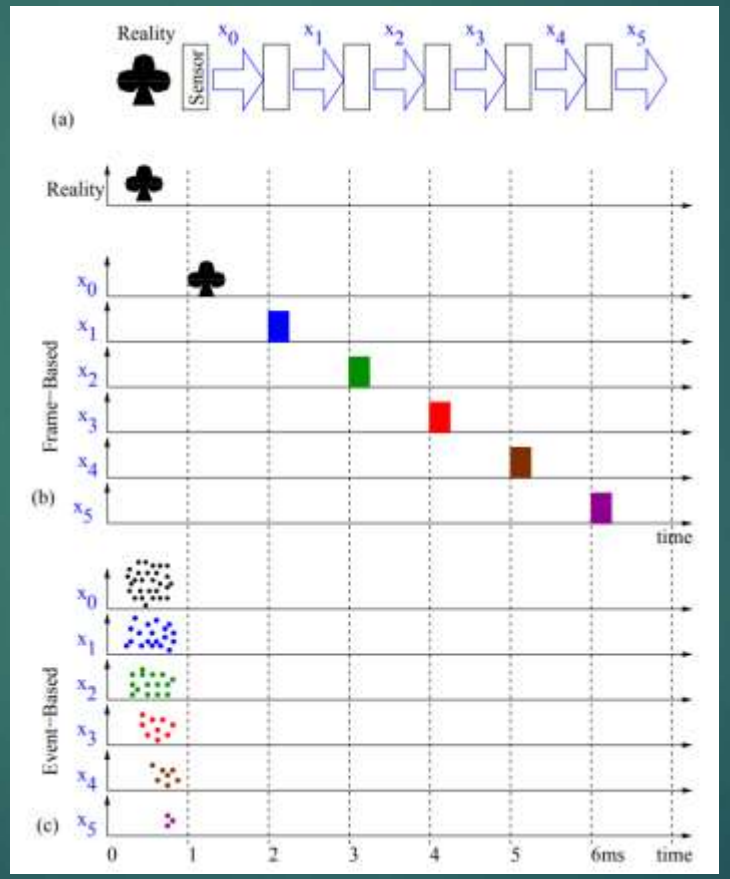
# Background-SNN

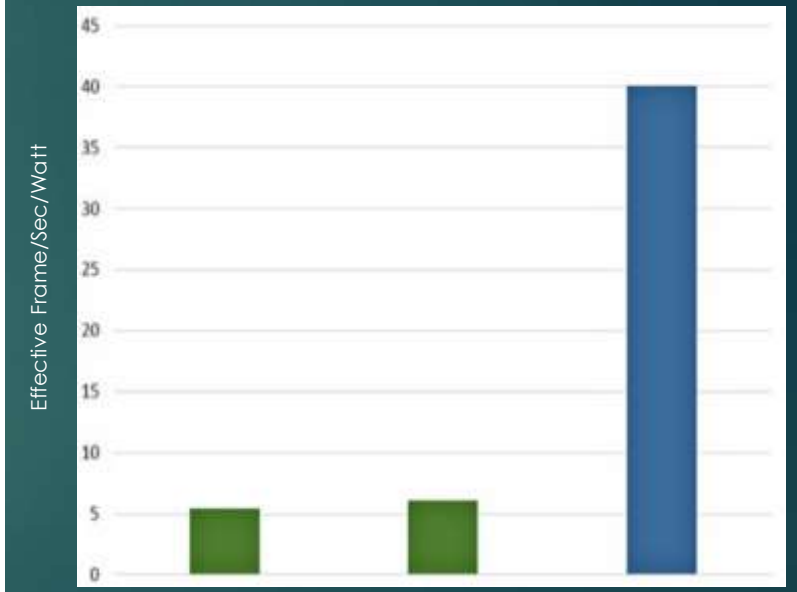





CNN vs SNN




CNN and SNN Speed Comparison  GPU and SNN Energy Comparison

# 3.1 Introduction



- Most existing SNN research uses rate-coded leaky integrate-and-fire neuron model that is not efficient to train
  - Discrete spikes are nondifferentiable
  - High computational complexity: have to solve differential equation for membrane potential during training
  - Highly Nonlinear input-output response leads to slow and ill convergence.
- Result: Most existing SNNs are limited to transfer learning, can not implement direct training.

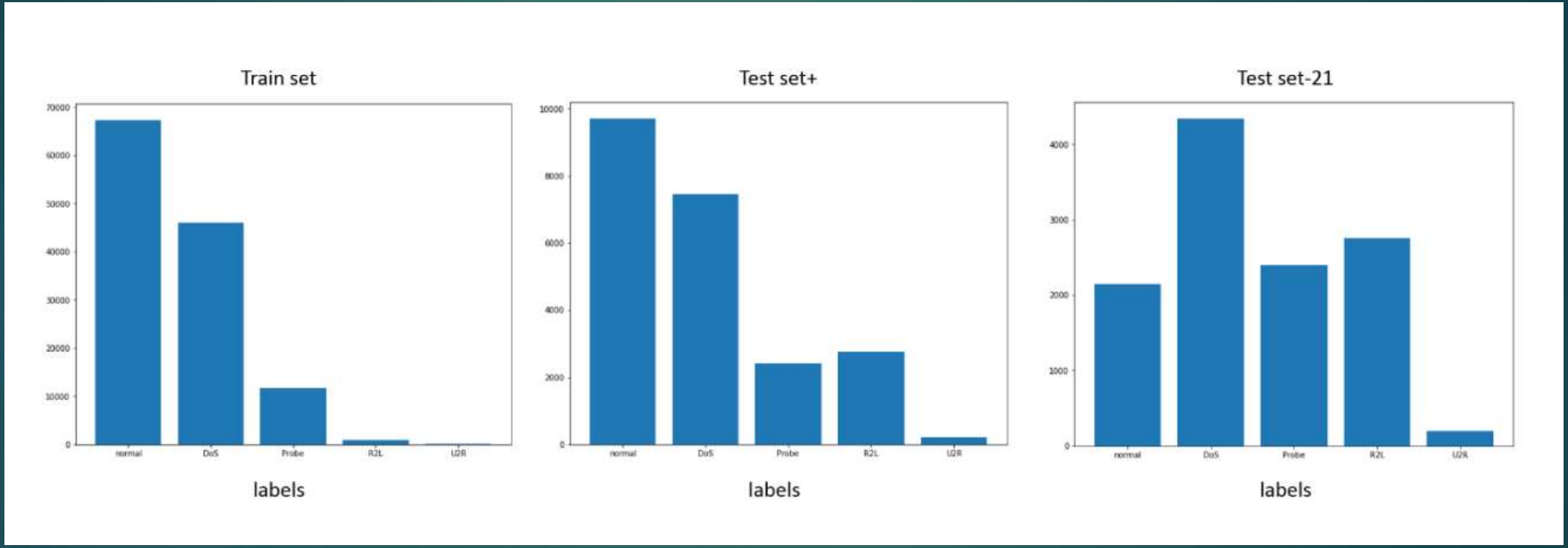
# Contribution



- Analyze the input-output response of SNN neurons to show that leaky neurons have a too complex and too nonlinear input-output response.
- Show that SNNs built with a special SNN neuron model can resolve these problems
- Demonstrate superior performance of such SNNs over two network intrusion datasets. This is the first time SNNs are train over these datasets.

# Network Intrusion Datasets-NSLKDD

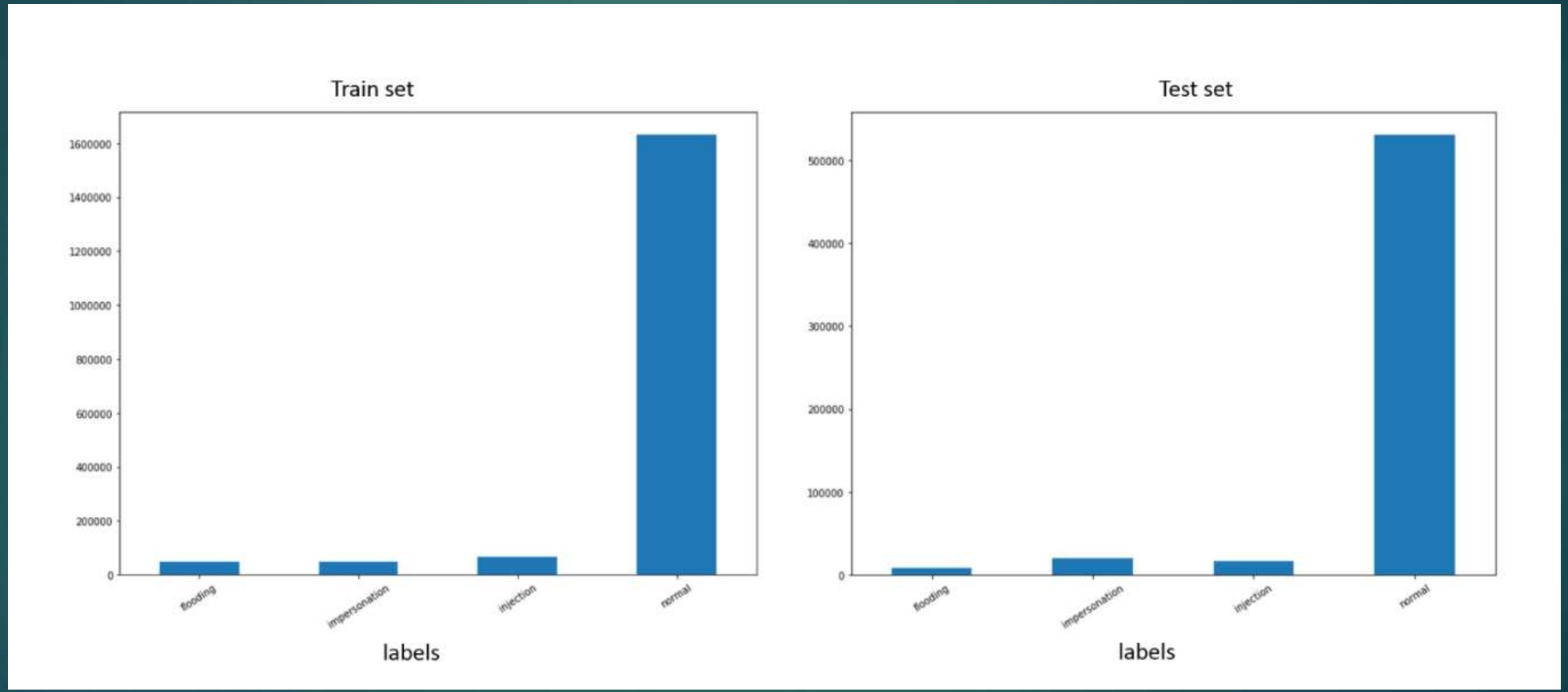





NSLKDD

# Network Intrusion Datasets-AWID






AWID

# Analyze Spiking Neuron Response

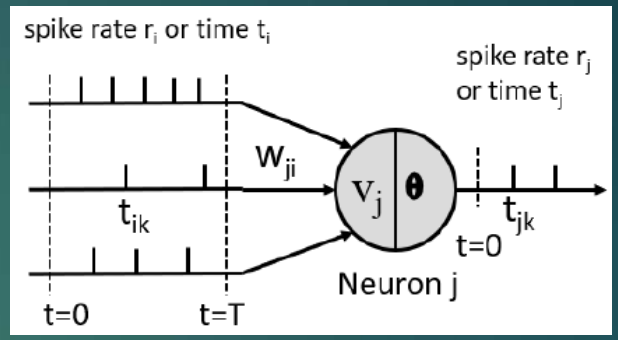


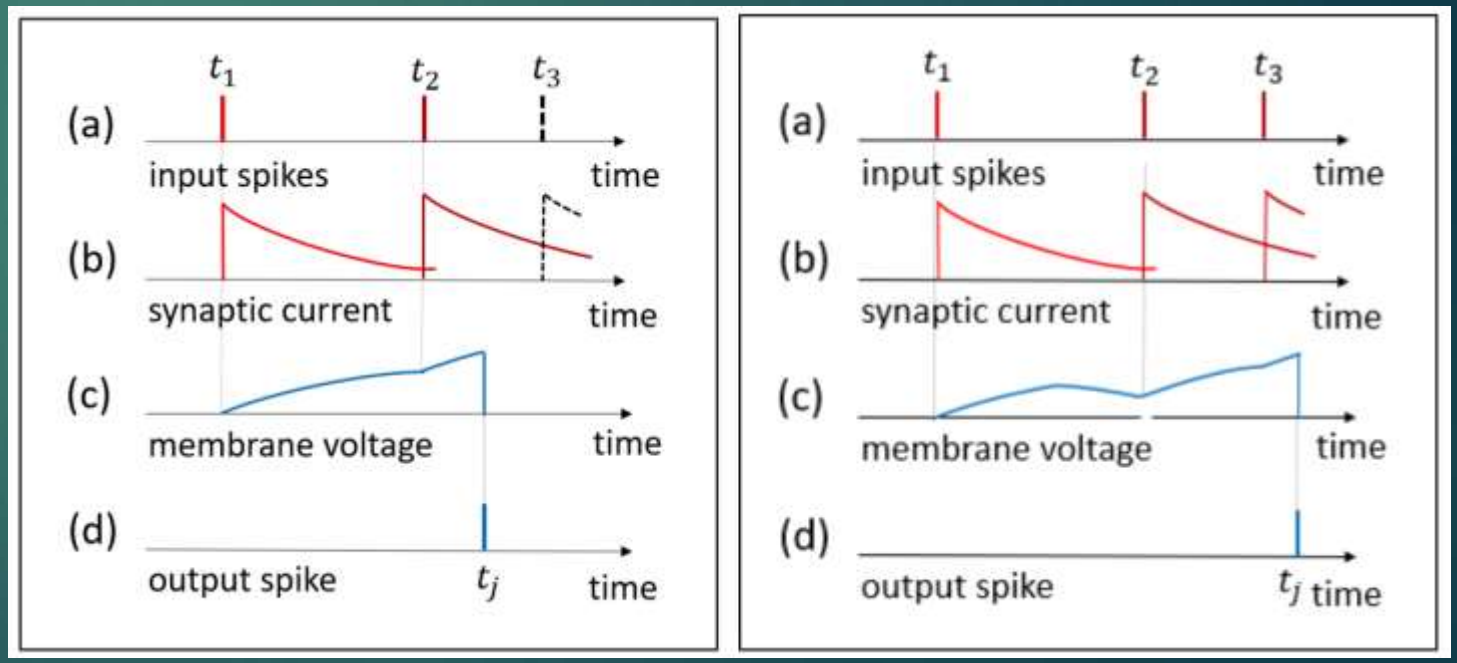
- N-LIF neuron membrane potential

$$\frac{dv_j(t)}{dt} = \sum_i w_{ji} g(t - t_i)$$

- LIF neuron membrane potential

$$\frac{dv_j(t)}{dt} + bv_j(t) = \sum_i w_{ji} g(t - t_i)$$




(i) n-LIF neuron

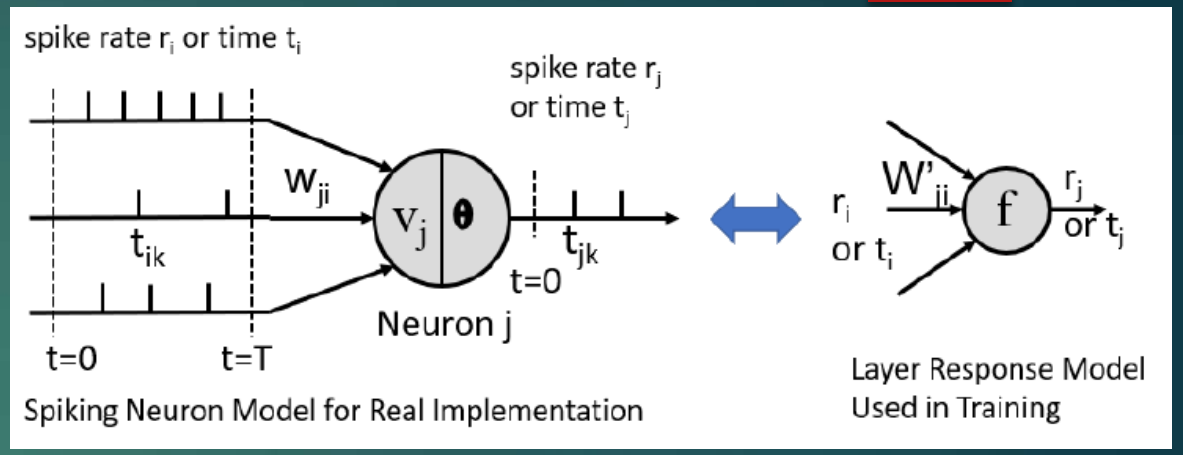
(ii) LIF neuron

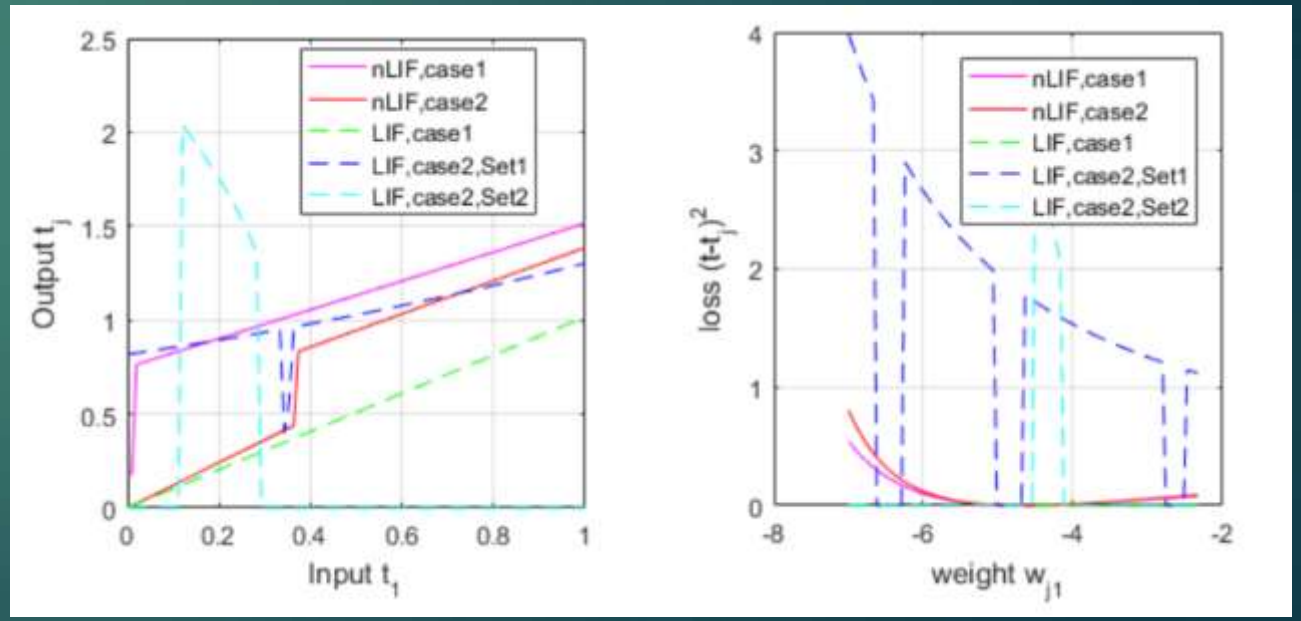
- Analyze input-output spike timing to derived closed-form expressions
  - LIF neuron has too complex and too nonlinear response
    - Not suitable to build deep SNNs

$$t_j = \frac{\sum_{i\in c} w_{ji} t_i e^{bt_i}}{\sum_{i\in c} w_{ji} e^{bt_i}} - \frac{1}{b} W\left(-\frac{bv_0}{\sum_{i\in c} w_{ji} e^{bt_i}} e^{\frac{b\sum_{i\in c} w_{ji} t_i e^{bt_i}}{\sum_{i\in c} w_{ji} e^{bt_i}}}\right)$$

  - N-LIF neuron has desirable response
    - Appropriate to build deep SNNs

$$e^{\frac{t_j}{\tau}} = \sum_{i\in c} e^{\frac{t_i}{\tau}} \frac{w_{ji}}{\sum_{l\in c} w_{jl} - \frac{v_0}{\tau}}$$




$t_j$ vs $t_i$.

Loss function $(t - t_j)^2$

# SNNs Built with n-LIF Neurons



- Develop SNNs with the single-spike temporal-coded n-LIF neuron's input-output response equation

$$e^{\frac{t_j}{\tau}} = \sum_{i \in C} e^{\frac{t_i}{\tau}} \frac{w_{ji}}{\sum_{l \in C} w_{jl} - \frac{v_0}{\tau}}$$

- Developed Algorithm 1 for training the SNNs

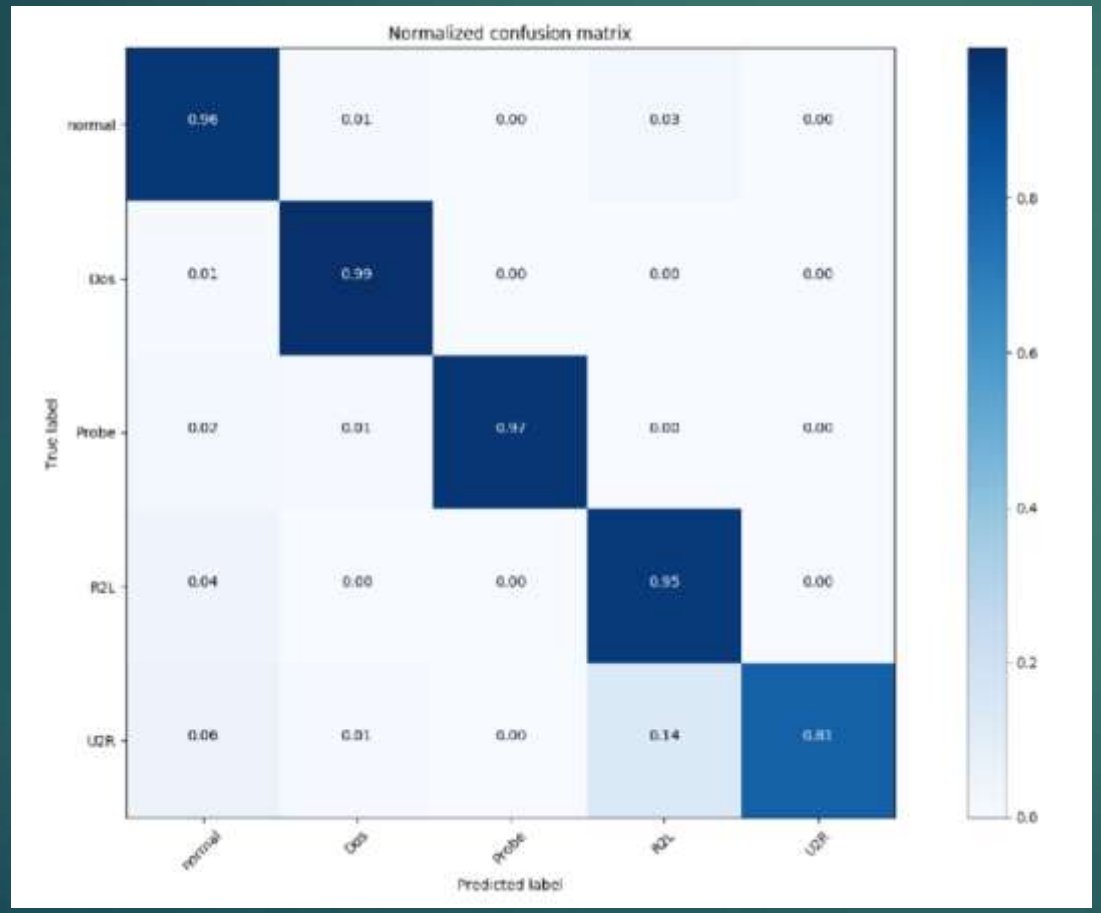
**Algorithm 1** Forward pass of a SNN neuron based on (11)

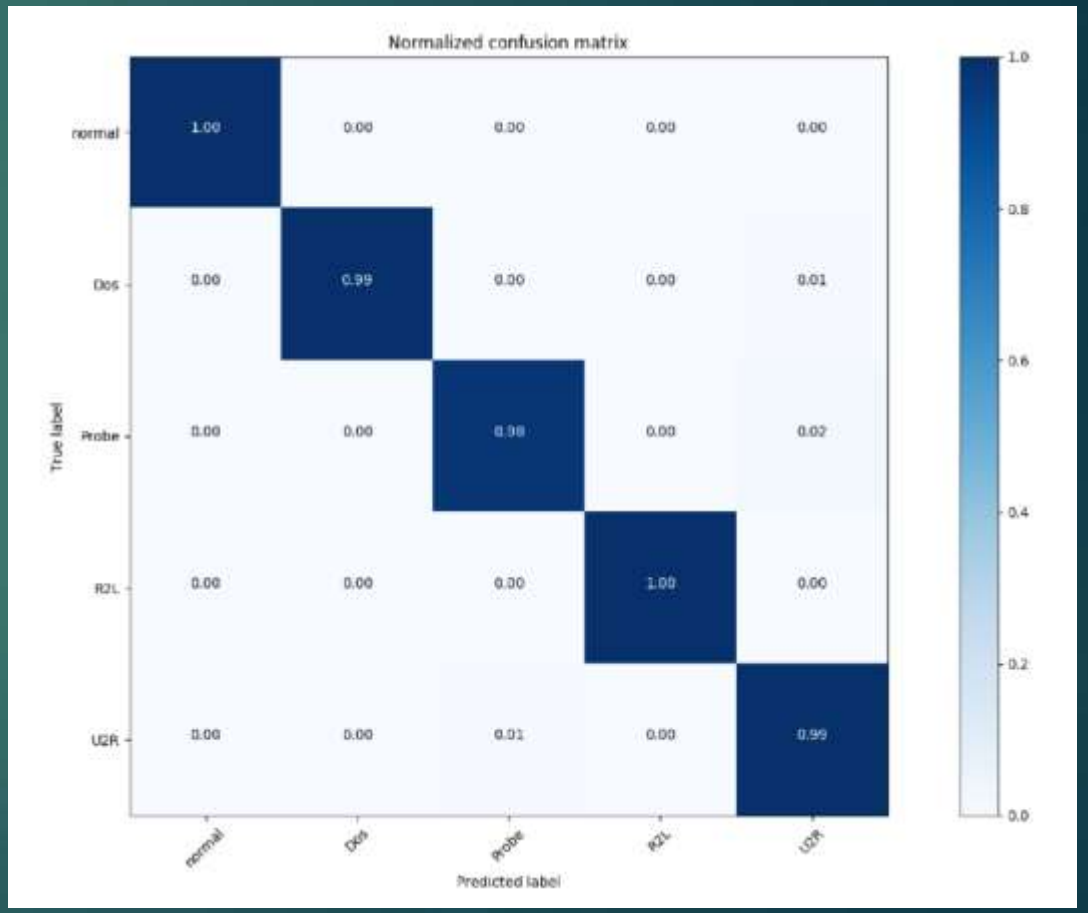
**Input:** $\mathbf{z} = [z_1, \cdots, z_N]$: input spiking time vector
**Input:** $\mathbf{w} = [w_1, \cdots, w_N]$: weight vector
**Output:** $z_{out}$: output spiking time
$\mathbf{i} \leftarrow argsort(\mathbf{z})$: ascending order index
$\mathbf{z}_{sorted} \leftarrow \mathbf{z}[\mathbf{i}]$: sorted input vector
$\mathbf{w}_{sorted} \leftarrow \mathbf{w}[\mathbf{i}]$: sorted weight vector
$\mathbf{z}_{cumsum} \leftarrow cumsum(\mathbf{z}_{sorted} * \mathbf{w}_{sorted})$: $\sum_{i \in C} w_{ji} z_i$
$\mathbf{w}_{cumsum} \leftarrow cumsum(\mathbf{w}_{sorted}) - v_0/\tau$: $\sum_{\ell} w_{j\ell} - v_0/\tau$
$\mathbf{z}_{candidate} \leftarrow \mathbf{z}_{cumsum} / \mathbf{w}_{cumsum}$: element-wise division
$\mathbf{z}_{shifted} \leftarrow [\mathbf{z}_{sorted}[2:], Inf]$: next input spiking time
$\mathbf{c} \leftarrow (\mathbf{z}_{candidate} > \mathbf{z}_{sorted}) \& (\mathbf{z}_{candidate} <= \mathbf{z}_{shifted})$
$k \leftarrow where(\mathbf{c} == True)[1]$: index of the first True element
**return** $z_{out} \leftarrow \mathbf{z}_{candidate}[k]$: output spiking time

# Experiments



- Experiment over NSL-KDD Dataset



# Experiments



- Experiment over NSL-KDD Dataset

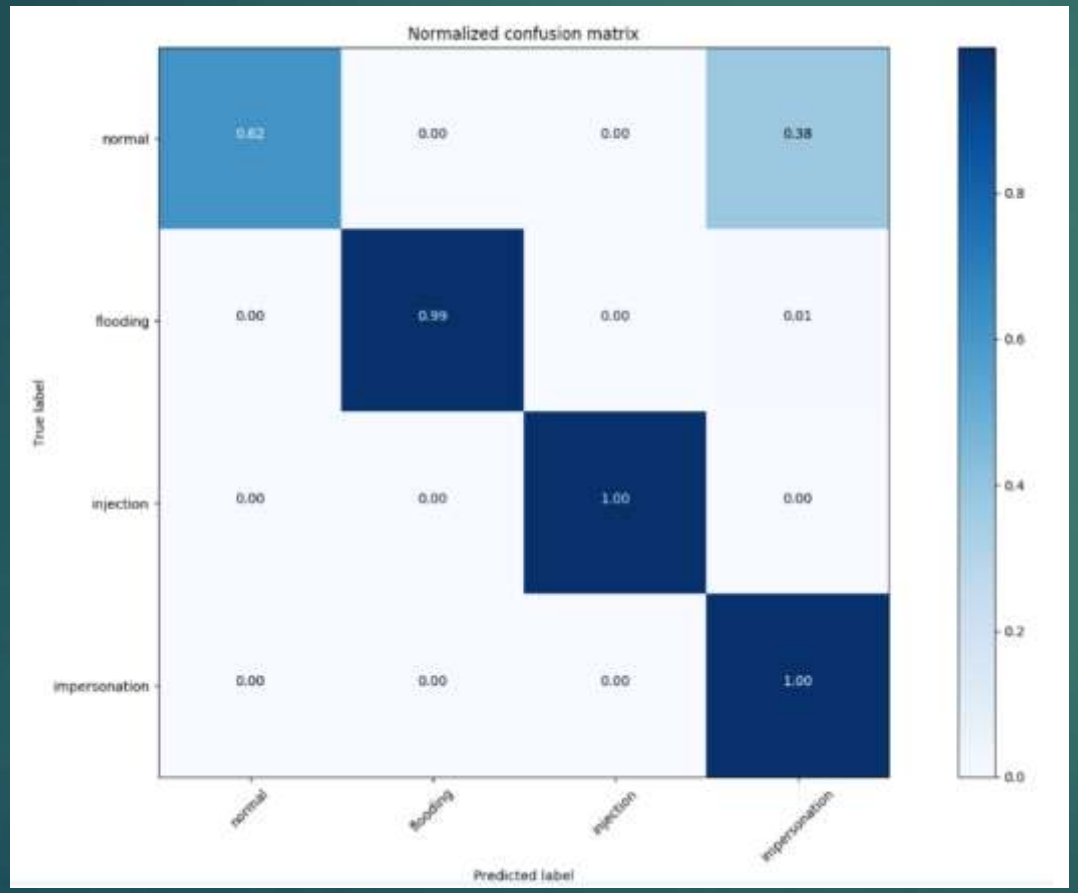
Our SNNs have the best performance (highest accuracy) than DNN/CNN, RL, and traditional matching learning methods

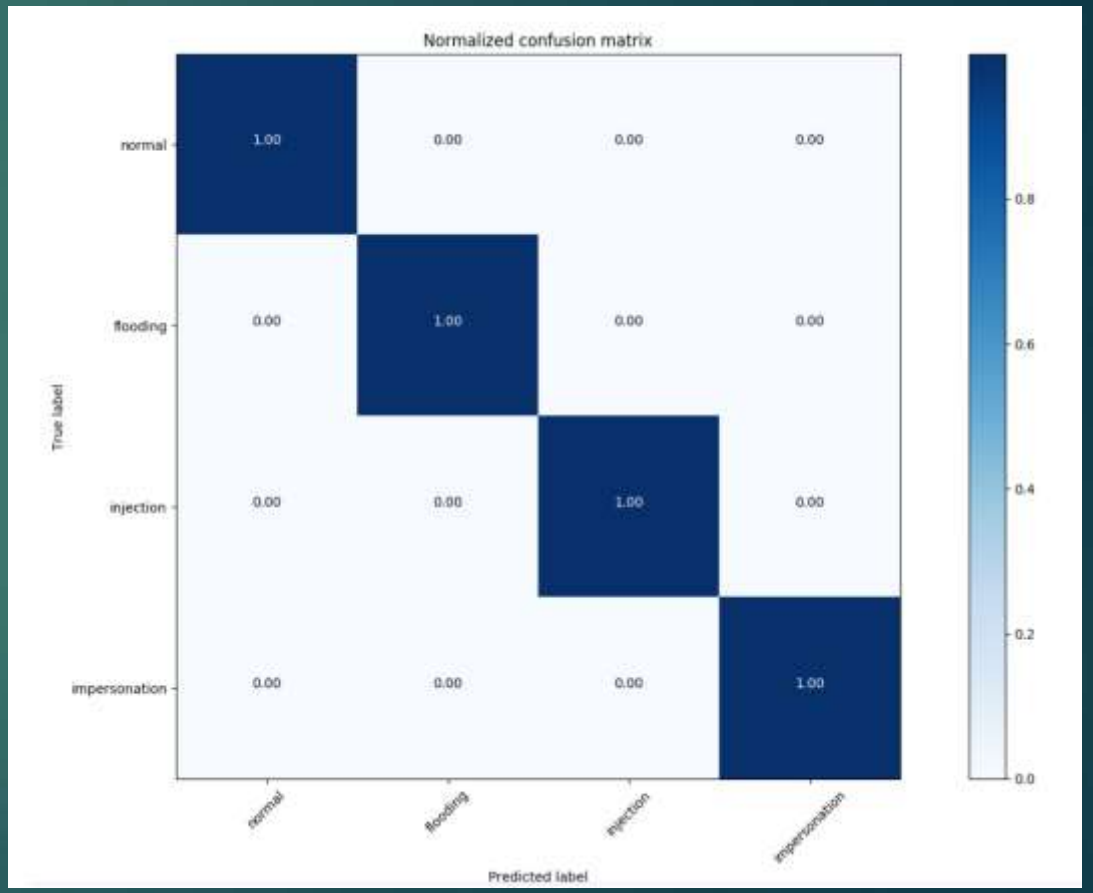
| | Accuracy | F1 | Precision | Recall |
|---|---|---|---|---|
| Logistic Regression | 0.7068 | 0.6807 | 0.8955 | 0.5491 |
| SVM | 0.8799 | 0.8927 | 0.9081 | 0.8779 |
| KNN | 0.7808 | 0.7769 | 0.9233 | 0.6706 |
| Random Forest | 0.7472 | 0.7211 | 0.9688 | 0.5743 |
| radient Tree Boosting | 0.7761 | 0.7612 | 0.9690 | 0.6267 |
| Naïve Bayes | 0.8019 | 0.7967 | 0.9583 | 0.6818 |
| AdaBoost | 0.7606 | 0.7403 | 0.9583 | 0.5992 |
| Neural Network | 0.7966 | 0.7881 | 0.9679 | 0.6647 |
| CNN-1D | 0.7875 | 0.7633 | 0.8094 | 0.7875 |
| Reinforcement Learn | 0.8978 | 0.9120 | 0.8944 | 0.9303 |
| Our DNN | 0.8834 | 0.8860 | 0.8936 | 0.8834 |
| Our CNN-1D | 0.9564 | 0.9561 | 0.9565 | 0.9564 |
| SNN (Original) | 0.9717 | 0.9718 | 0.9721 | 0.9717 |
| SNN (Resampled) | **0.9931** | **0.9931** | **0.9931** | **0.9931** |

# Experiments



- Experiment over AWID Dataset



# Experiments



- Experiment over AWID Dataset

Our SNNs have the best performance (highest accuracy) than DNN/CNN, RL, and traditional matching learning methods

| | Accuracy | F1 | Precision | Recall |
|---|---|---|---|---|
| AdaBoost | 0.9220 | 0.8850 | 0.8500 | 0.9220 |
| Decision Tree | 0.9620 | 0.9480 | 0.9620 | 0.9630 |
| Naïve Bayes | 0.9055 | 0.9090 | 0.9170 | 0.9060 |
| Frequency Tabel | 0.9457 | 0.9220 | 0.9000 | 0.9460 |
| Random Forest | 0.9582 | 0.9440 | 0.9590 | 0.9580 |
| Neural Network | 0.9470 | 0.9256 | 0.9174 | 0.9473 |
| Reinforcement Learn | 0.9570 | 0.9394 | 0.9235 | 0.9570 |
| Our DNN | 0.9585 | 0.9624 | 0.9715 | 0.9585 |
| Our CNN-1D | 0.9528 | 0.9351 | 0.9504 | 0.9528 |
| SNN (Original) | 0.9898 | 0.9893 | 0.9895 | 0.9898 |
| SNN (Resampled) | **0.9984** | **0.9985** | **0.9985** | **0.9984** |

# Conclusions



- A comprehensive study of SNNs and their application in network intrusion detection
  - Analyze the input-output response of spiking neurons to find the best one to build direct-train SNNs
  - Apply single-spike temporal-coded n-LIF neuron to develop direct-train SNN framework
  - Develop SNNs for network intrusion datasets, and show the SNNs outperformed a list of existing methods including the DNN-based methods

# Thanks !