Spiking Neural Network (SNN) Design and Analysis for MNIST Classification

Introduction –

Spiking Neural Networks (SNNs) are the third generation of neural networks, designed to more closely mimic the event-driven behavior of biological neurons. Unlike conventional artificial neural networks (ANNs), SNNs process information in the form of discrete spike events, enabling low-power, highly efficient computation that is attractive for neuromorphic hardware.

This project focuses on the design, training, optimization, and evaluation of an SNN for classifying images from the MNIST handwritten digit dataset. The network processes spike-encoded image data and outputs spike-based classifications. The study includes both supervised learning using surrogate gradient descent and unsupervised learning using spike-timing-dependent plasticity (STDP).

The main objectives of this project are:

- To design and train a multi-layer spiking neural network for MNIST digit classification.

- To preprocess MNIST images into various resolutions (4×4, 7×7, 14×14, and 28×28) and encode them into spike trains using Poisson encoding.

- To explore supervised learning (surrogate gradient descent) and unsupervised learning (STDP) methods for training SNNs.

- To implement weight quantization at different bit levels (2-bit to 5-bit) and evaluate its impact on accuracy.

- To design a smaller, hardware-efficient SNN model achieving high classification accuracy (>75%).

- To visualize spike activity through raster plots for different network layers.

- To compare performance, training methods, and network designs based on accuracy, complexity, and hardware suitability.

Methodology

Network Design –

The spiking neural network (SNN) designed in this project consists of a fully connected feedforward architecture. It includes an input layer, two hidden layers, and an output layer corresponding to the 10 digit classes (0–9).

The main architecture used during initial experiments and training was as follows:

- Input Layer: Resized MNIST images (sizes: 4×4, 7×7, 14×14, or 28×28 pixels)

- Hidden Layer 1: 512 neurons

- Hidden Layer 2: 256 neurons

- Output Layer: 10 neurons

Each neuron integrates incoming spikes over time and generates output spikes using the Leaky Integrate-and-Fire (LIF) model with surrogate gradient-based approximations during backpropagation.

Later, for hardware efficiency, a **smaller network** was also designed with the following architecture:

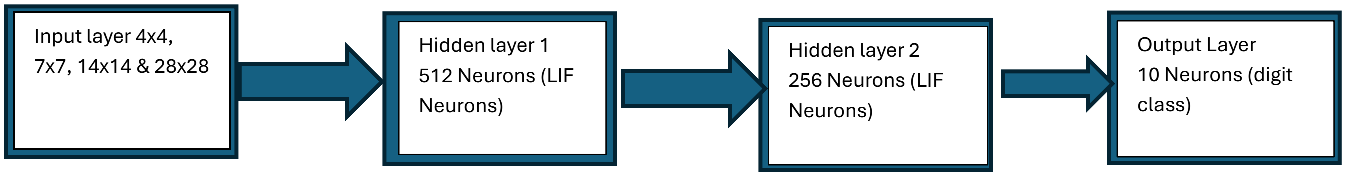
- Hidden Layer 1: 128 neurons

- Hidden Layer 2: 64 neurons

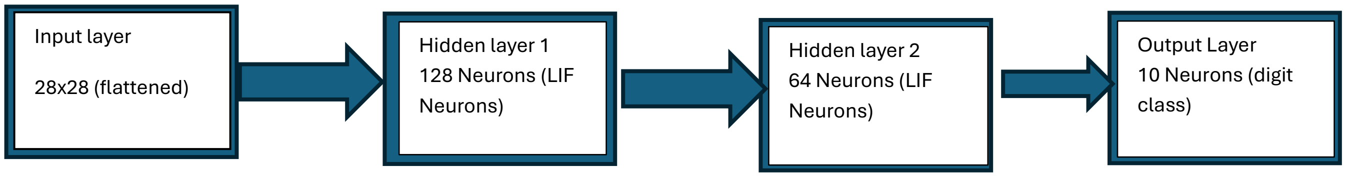
- Output Layer: 10 neurons

This smaller network still achieved a high test accuracy of 97.58%, making it highly suitable for low-power neuromorphic deployment.

The figure below summarizes the general architecture of the SNN:


Input layer 4x4, 7x7, 14x14 & 28x28
Hidden layer 1 512 Neurons (LIF Neurons)
Hidden layer 2 256 Neurons (LIF Neurons)
Output Layer 10 Neurons (digit class)

Smaller network


Input layer 28x28 (flattened)
Hidden layer 1 128 Neurons (LIF Neurons)
Hidden layer 2 64 Neurons (LIF Neurons)
Output Layer 10 Neurons (digit class)

<u>Image Processing –</u>

To prepare the MNIST dataset for spiking neural network (SNN) training, the original 28×28 grayscale images were resized to smaller dimensions: 4×4, 7×7, and 14×14 pixels, in addition to retaining the original 28×28 size.

The resizing was done using bilinear interpolation, ensuring that important spatial features were preserved while significantly reducing the number of input neurons for hardware-friendly designs.

This preprocessing step allowed the evaluation of how different input resolutions affect SNN performance, energy consumption, and memory requirements.

<u>Spike Encoding –</u>

After resizing, the images were converted into spike trains using Poisson spike encoding. In this method, the pixel intensity of each image is treated as the firing probability of a neuron at each time step.

For each pixel, a random number is generated at every time step and compared to the normalized pixel value. If the random number is less than the pixel value, the neuron emits a spike (value 1); otherwise, it remains silent (value 0). This probabilistic encoding simulates the variability of biological neurons.

To study the temporal behavior of the spiking network, the encoded data was generated over different simulation time steps: 5, 10, and 20.

- Shorter time steps (5) result in faster inference but reduced time-place richness.

- Longer time steps (20) allow better spike accumulation and feature extraction, potentially improving classification accuracy at the cost of increased latency.

- Intermediate time steps (10) provided a balance between performance and efficiency.

Overall, Poisson encoding effectively transformed static MNIST images into dynamic spike streams suitable for SNN processing.

Training Methods –

Two different learning methods were implemented and compared for training the spiking neural network (SNN):

1. Surrogate Gradient Descent (Supervised Learning)

   Since spike generation is a non-differentiable operation, traditional backpropagation cannot be directly applied to SNNs. To address this, surrogate gradient methods were used.

   In surrogate gradient descent (SGD):

   - The spiking nonlinearity is replaced during backpropagation by a smooth, differentiable surrogate function (e.g., a sigmoid or piecewise linear function).

   - The SNN was trained using a supervised cross-entropy loss based on the network's output at the final time step.

   - The Adam optimizer was used with a learning rate of 1e-3.

- The network was trained for 5–10 epochs depending on the image size and architecture.

- After each batch, neuron membrane potentials were reset to simulate biologically plausible resets.

This method enabled effective supervised training of the SNN, achieving high classification accuracy.

2. Spike-Timing Dependent Plasticity (Unsupervised Learning)

Spike-timing-dependent plasticity (STDP) is a biologically inspired unsupervised learning rule where the synaptic weight changes are based on the relative timing of presynaptic and postsynaptic spikes:

- If a presynaptic neuron fires shortly before a postsynaptic neuron, the synapse is strengthened (Long-Term Potentiation, LTP).

- If the presynaptic neuron fires after the postsynaptic neuron, the synapse is weakened (Long-Term Depression, LTD).

In this project:

- A custom simple Hebbian STDP learning rule was implemented manually because the standard SpikingJelly STDP module was not available.

- No labels were used during STDP training; learning was driven purely by spike activity.

- The STDP network was trained on 14×14 MNIST inputs.

- STDP training converged after approximately 5 epochs.

Although STDP learning did not directly produce high classification accuracy like supervised SGD, it demonstrated the SNN's ability to self-organize features without supervision, making it attractive for energy-efficient hardware.

Weight Quantization –

To further optimize the trained spiking neural networks (SNNs) for hardware implementation, weight quantization was performed.

Weight quantization involves reducing the precision of network weights from full floating-point (32-bit) representations to lower-bit fixed-point representations. In this project, weights were quantized to 2-bit, 3-bit, 4-bit, and 5-bit levels.

The quantization process was as follows:

- For each trained weight, the range was divided into discrete levels depending on the bit width.
- A uniform linear quantization approach was used to map weights to the nearest quantized level.
- After quantization, the network was evaluated without retraining to measure accuracy degradation.

The motivation for quantization is to significantly reduce memory usage, simplify arithmetic operations, and enable efficient deployment on neuromorphic hardware where low-precision computation is preferred.

Results showed that:

- 3-bit, 4-bit, and 5-bit quantization preserved high classification accuracy (>90%).

- 2-bit quantization led to some performance degradation, especially on smaller networks or lower time step settings.

Thus, weight quantization offers a favorable tradeoff between accuracy and hardware resource efficiency.

Accuracy and performance evaluation

The trained spiking neural network (SNN) was evaluated across multiple configurations using different input image sizes, time steps, training methods, and weight quantization levels.

1. Effect of Image Size and Time Steps (SGD Training)

The following table summarizes test accuracy (%) for different combinations of image size and time steps using surrogate gradient descent:

Image Size	**Time Steps**	**Test Accuracy**
4x4	5	26.26%
4x4	10	34.58%
4x4	20	46.61%
7x7	5	66.49%
7x7	10	76.16%
7x7	20	84.90%
14x14	5	89.77%
14x14	10	93.05%
14x14	20	94.99%
28x28	5	94.46%
28x28	10	95.94%
28x28	20	95.43%

- As the **image size increases**, accuracy improves due to richer spatial features.
- **Increasing time steps** enhances temporal integration but at the **cost of inference latency.**
- Best performance was achieved with 28×28 input and 10 time steps.

2. Effect of weight quantization –

Bit Width	**Test Accuracy**
2 bit	90.52%
3 bit	95.85%
4 bit	95.73%
5 bit	96.09%

- Quantization to **3-bit or higher** retained nearly full accuracy.
- **2-bit quantization** caused a drop in performance but remained above 90%.

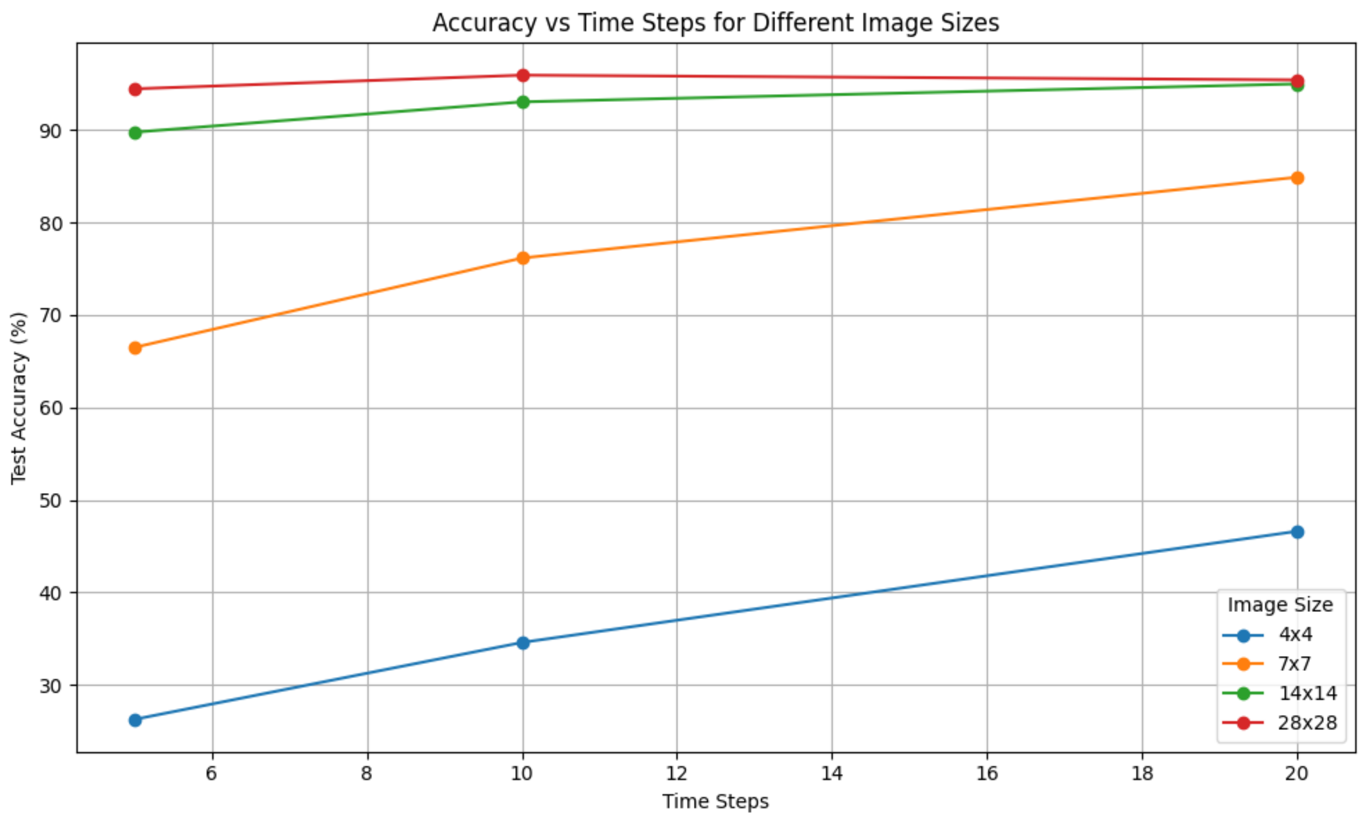
3. Comparison of training methods –

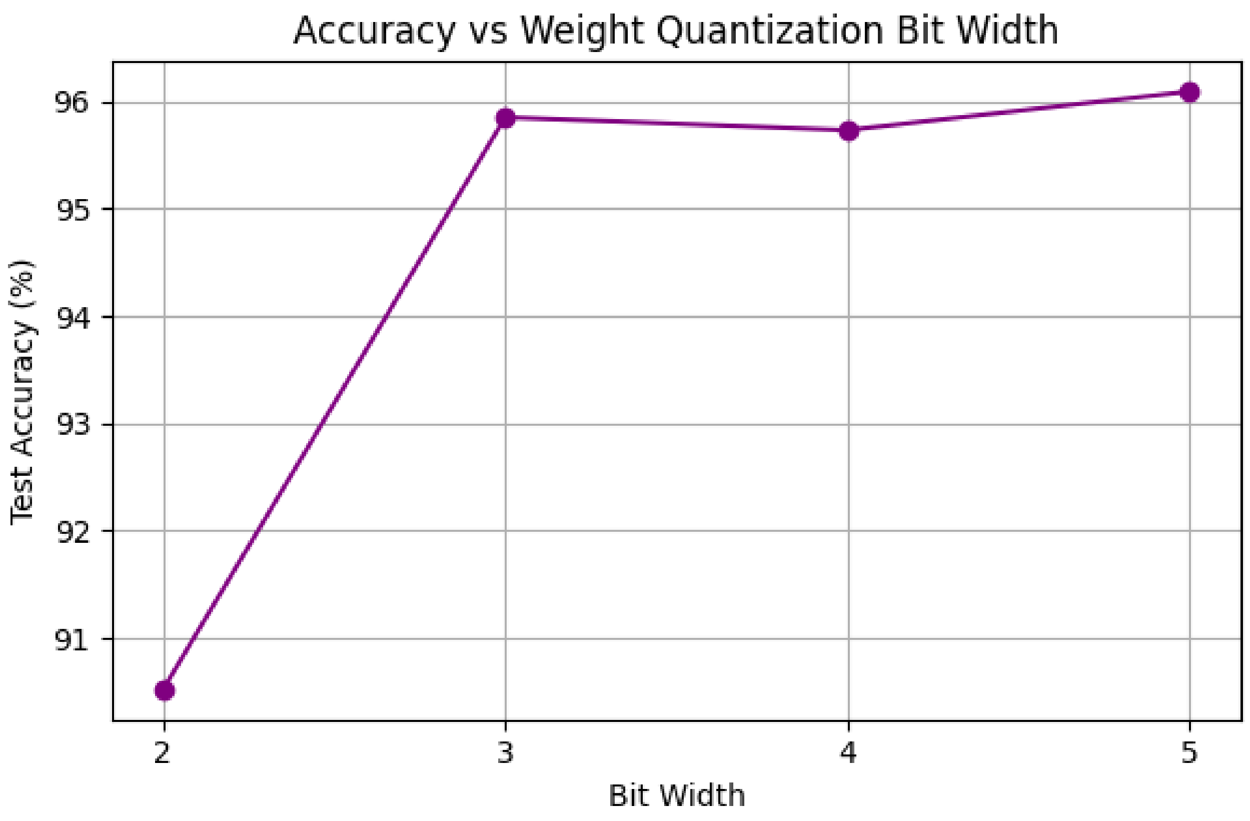
Training methods	Accuracy	Labels Used	Training Type
SGD (28x28, 10 ts)	95.94%	Yes	Supervised
STDP (14x14)	N/A	No	Unsupervised

- SGD provides high accuracy but requires labeled data and computational backpropagation.
- STDP is label-free and hardware-friendly but lacks strong classification accuracy alone.

4. Inference Latency –

- Time steps directly determine latency:
    - 5-time steps = Fast inference
    - 10-time steps = Balanced accuracy and latency
    - 20-time steps = Higher accuracy, slower speed
- For most experiments, **10-time steps** was the optimal trade-off.





Network Optimization –

To optimize the network for hardware deployment, a smaller SNN architecture was designed by reducing the number of hidden layer neurons while maintaining acceptable accuracy.

1. Final Smaller Network Architecture

Layer	**Neurons**
Input Layer	28x28
Hidden Layer 1	128
Hidden Layer 2	64
Output Layer	10

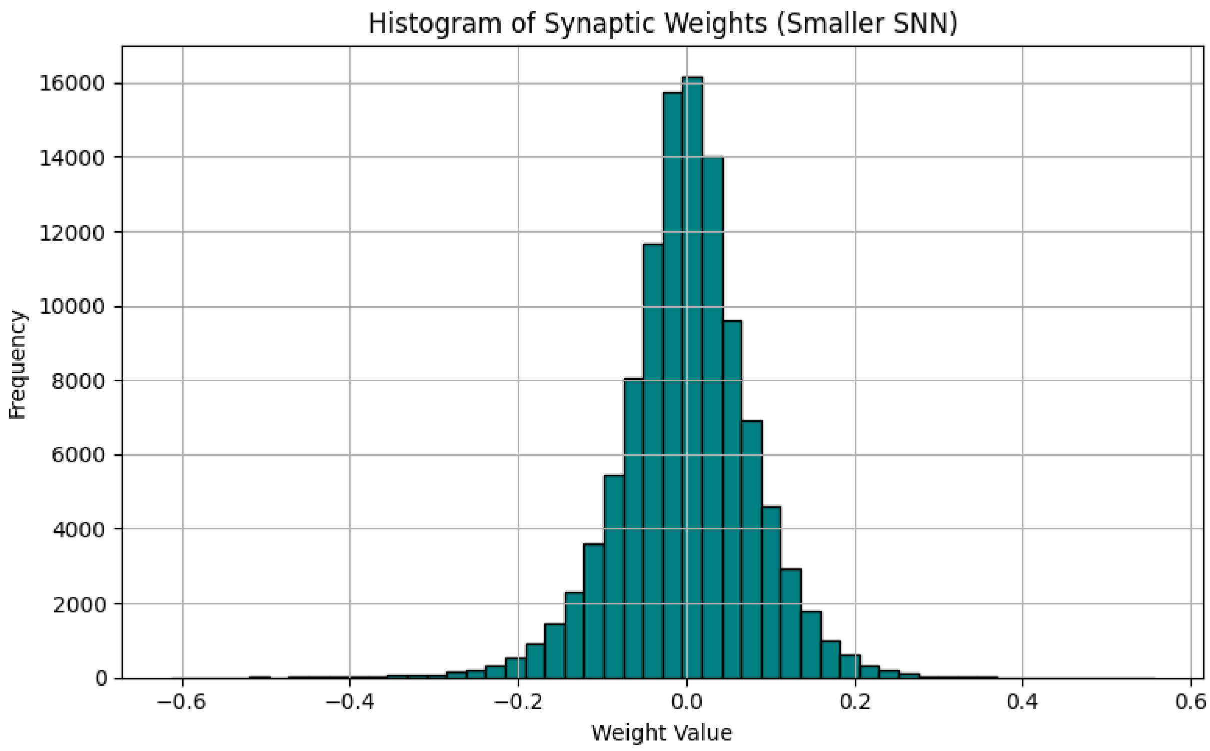
- The number of neurons in each hidden layer was reduced from 512 and 256 to 128 and 64 respectively.
- The model was trained using surrogate gradient descent (SGD) for 10 epochs on full-resolution 28×28 MNIST inputs with 10-time steps.

2. Performance of smaller network

    - Final training accuracy: **98.69%**
    - Final test accuracy: **97.58%**

    This test accuracy far exceeds the required target of 75%, demonstrating that the optimized network maintains excellent classification performance while significantly reducing complexity and memory usage.

3. Synaptic Weight Histogram
    The distribution of trained synaptic weights was visualized using histograms. Below is the histogram for the final trained smaller model:



The majority of weights are concentrated near zero, indicating efficient and stable training without saturation.

Spike Activity Visualization –

To gain insights into the internal behavior of the spiking neural network (SNN), spike raster plots were generated for the best-performing smaller network when classifying digits **6, 8, and 0**.

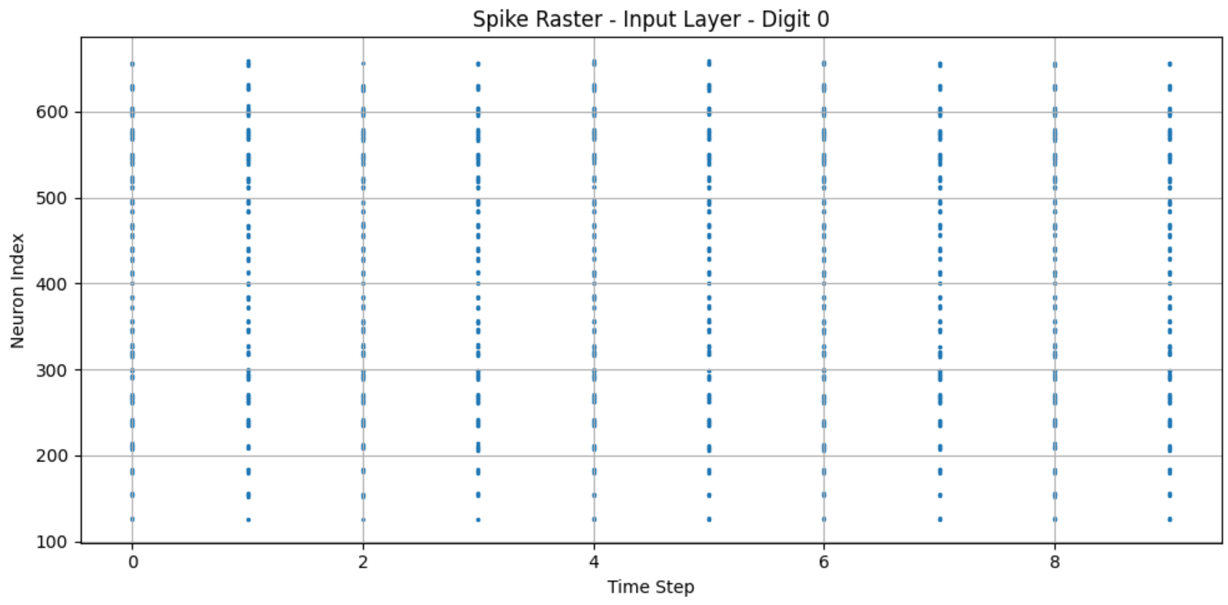
A spike raster plot shows the timing and activity of neurons in each layer over a series of time steps. Each dot represents a spike emitted by a neuron at a specific time step.

1. Input Layer (Poisson Encoder)

Observation: The input raster shows dense vertical lines because Poisson spike encoding emits spikes probabilistically based on pixel intensity.

Interpretation: Brighter pixels (e.g., parts of the digit curve) produce more frequent spikes. The spatial layout is preserved through time, providing rich temporal input.

Figure: Input layer spike raster for digit 0

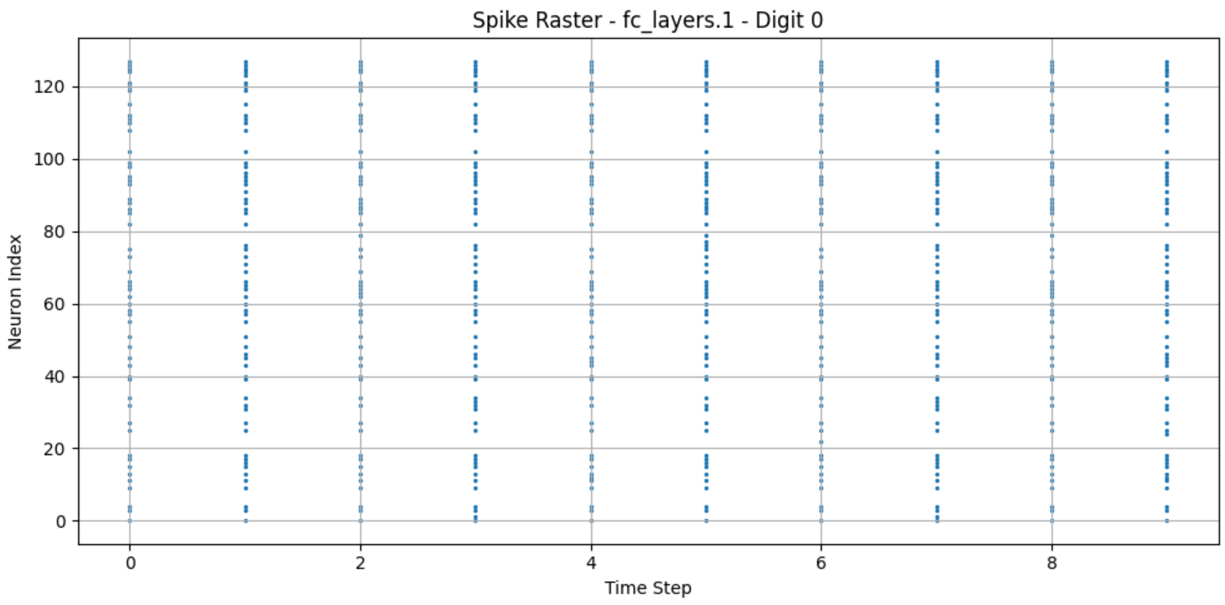


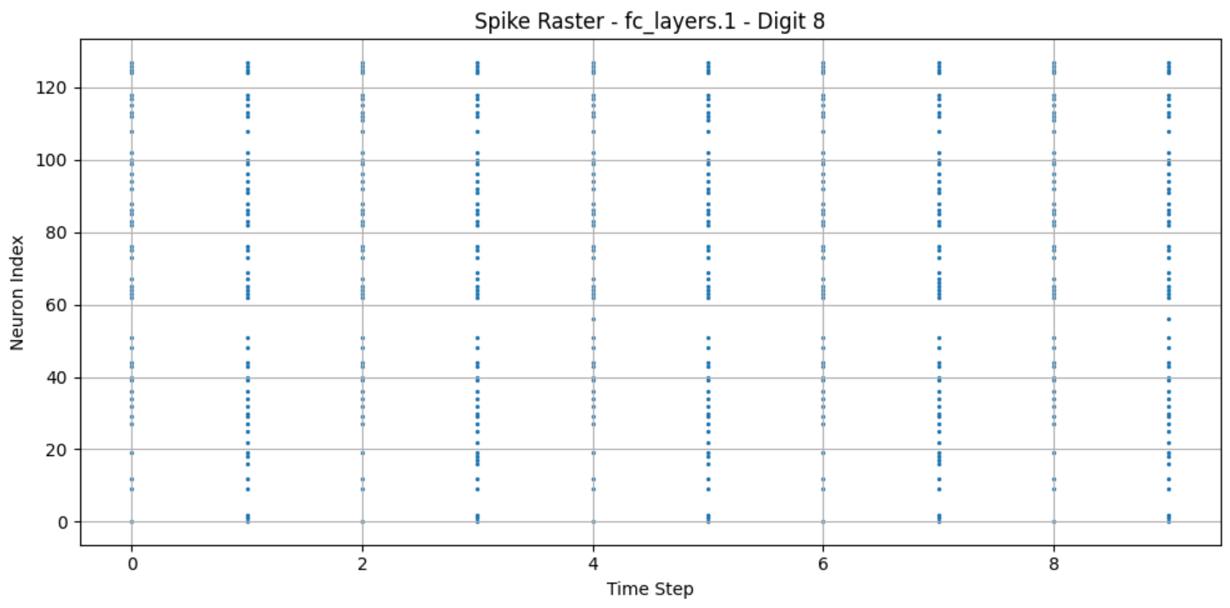
2. Hidden Layer 1 (fc_layer.1)

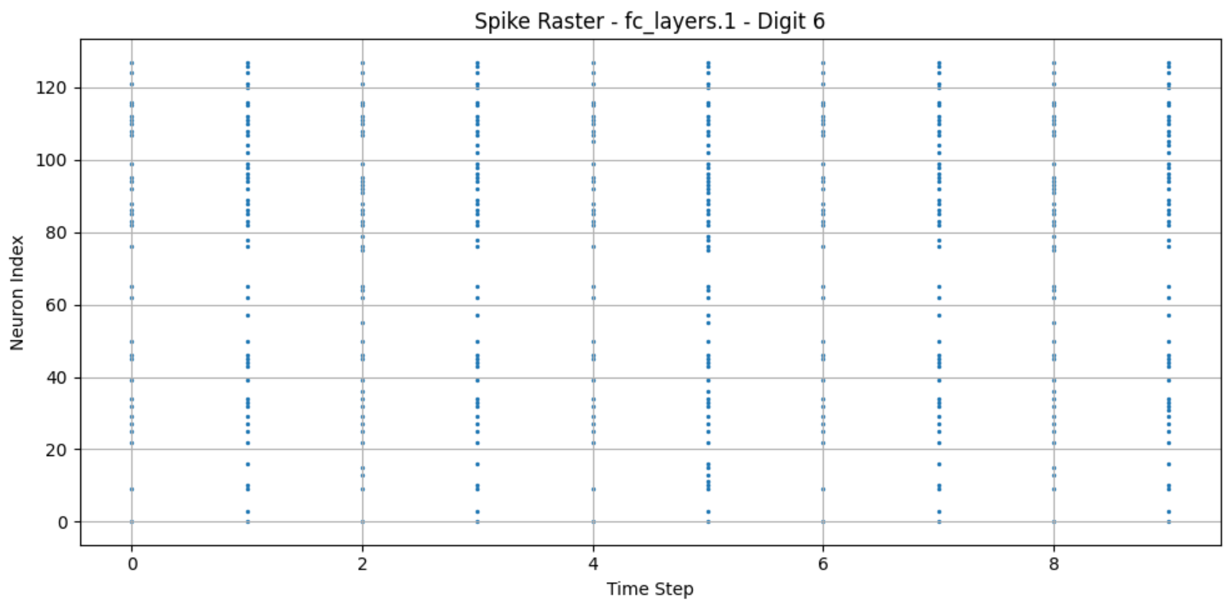
Observation: This layer shows moderate to dense spiking across many neurons for all three digits. The spikes are distributed across all time steps.

Interpretation: This suggests that neurons in this layer are actively extracting intermediate features from the temporally encoded input. A subset of neurons consistently spike for specific digits, indicating learned digit-specific patterns.

Figures – Input layer 1 Digit 0, 6, 8





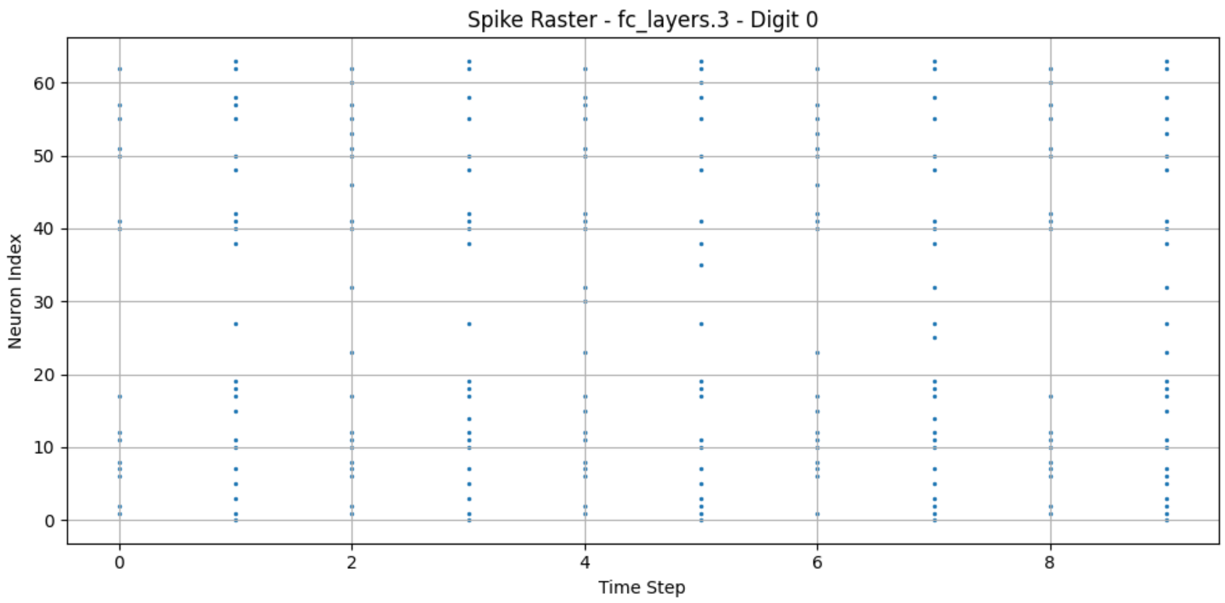


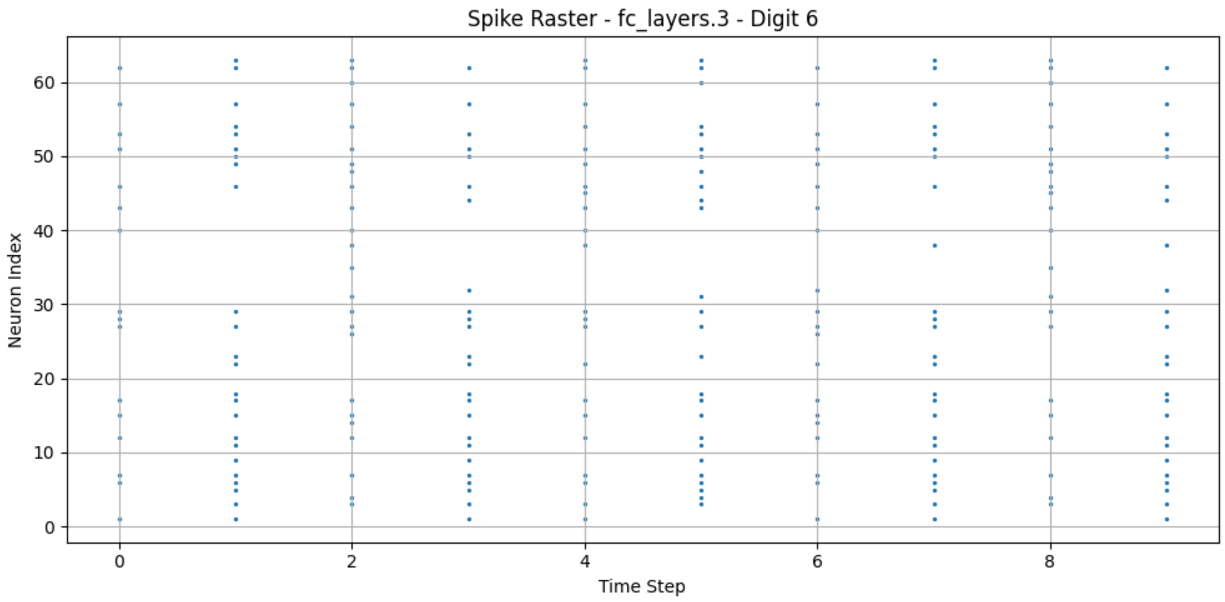
3. Hidden Layer 2 (fc_layer.3)

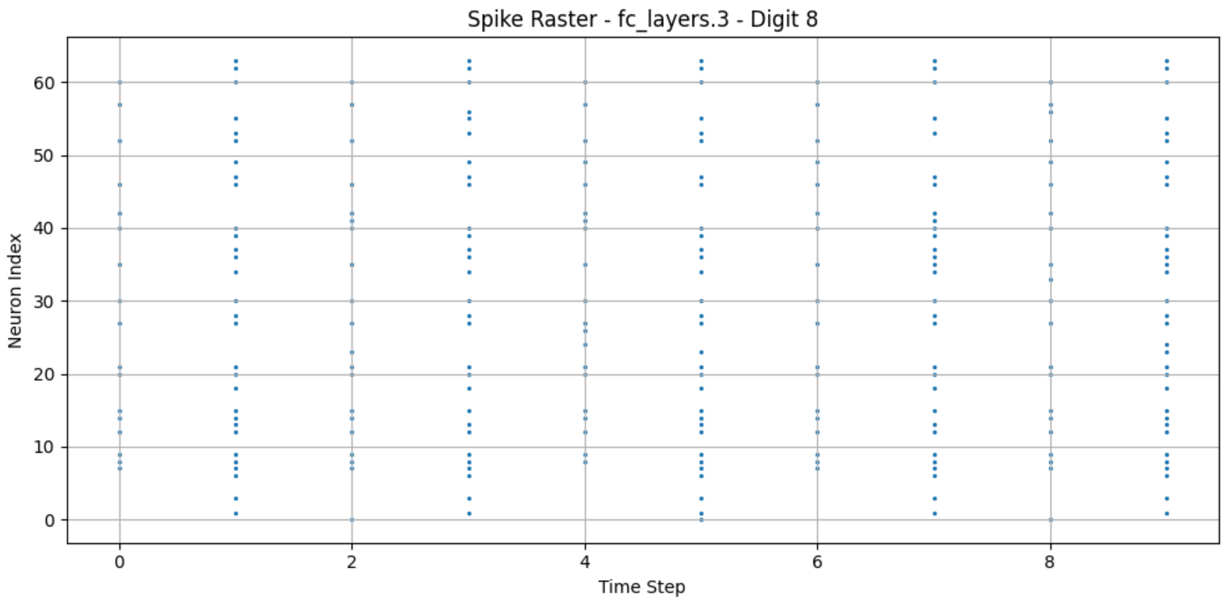
Observation: The spiking activity becomes more sparse and focused in this layer. Only a limited number of neurons are active at each time step.

Interpretation: The SNN is becoming more selective, reducing redundancy and forwarding only the most salient features to the output layer. Certain neurons appear to specialize in detecting parts of digits (e.g., the loop in ‘8’ or the curve in ‘6’).

Figures – Input layer 2 Digit 0, 6, 8





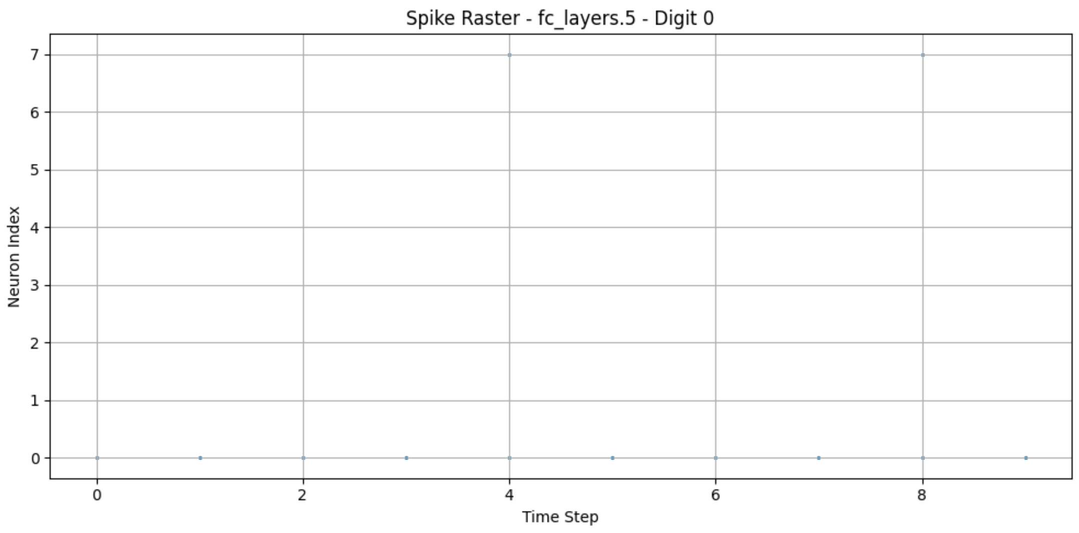


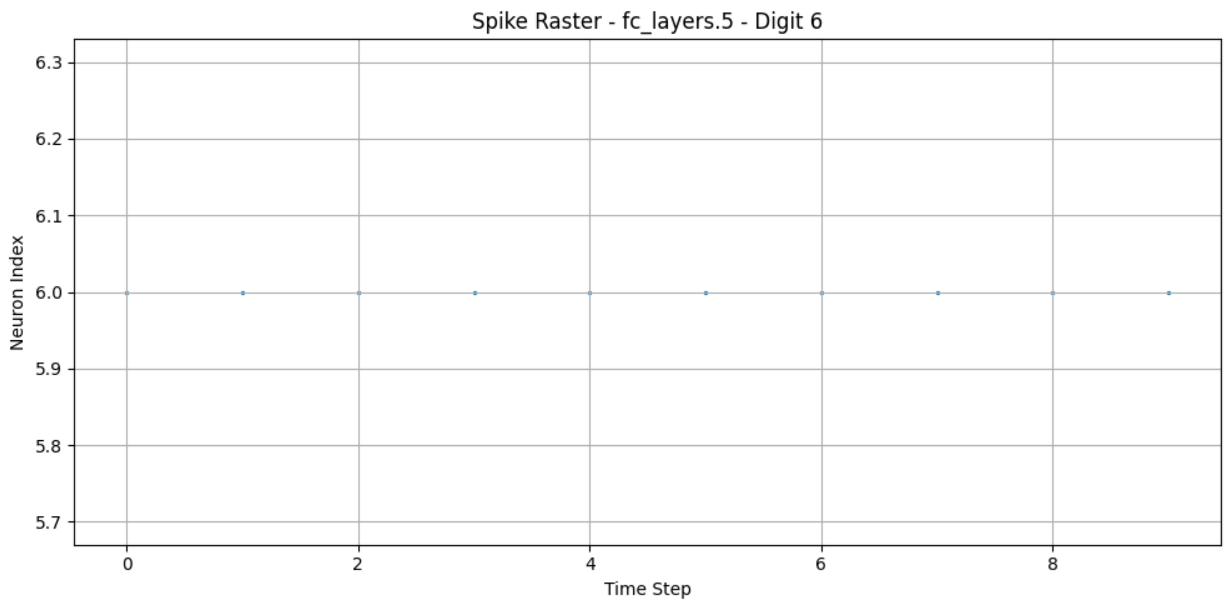
4. Output Layer (fc_layer.5)

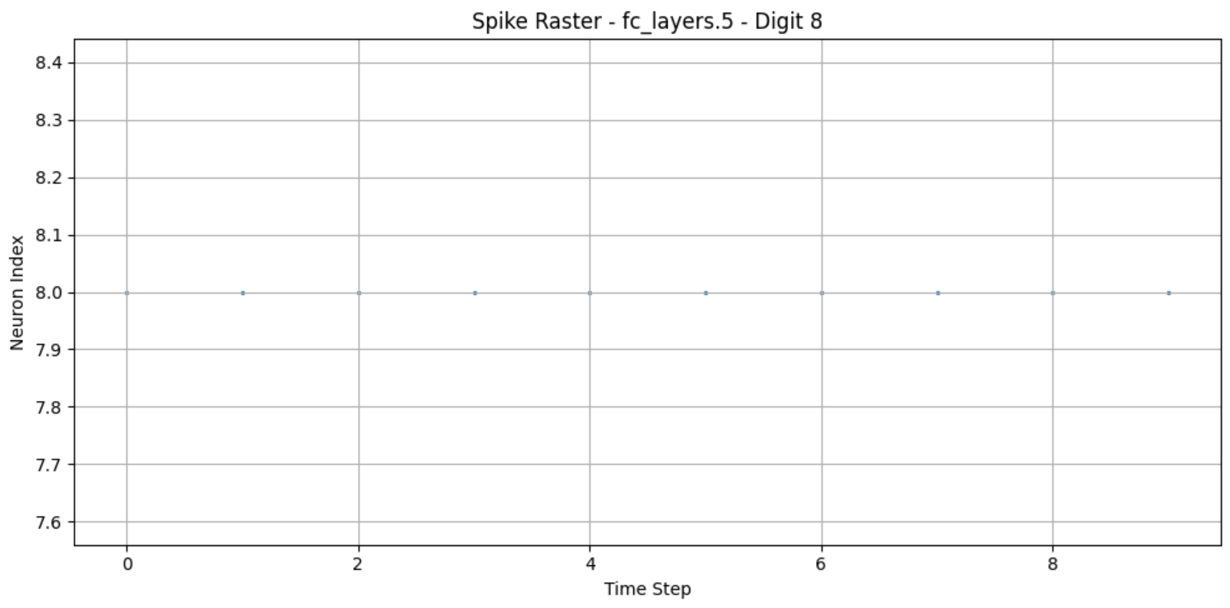
   Observation: For each digit, only one or two neurons spike across time, corresponding to the correct digit label (e.g., neuron 6 spikes for digit 6).

   Interpretation: This final layer acts as a classifier. The sparse output confirms that the network has confidently decided, with minimal ambiguity. The consistent activation of the correct output neuron across time confirms successful training.

   Figures are as follows –







Digit	**Most Active Output Neuron**	**Comments**
0	Neuron 0 and sometimes 7	Minor ambiguity, but neuron 0 dominates
6	Neuron 6	Consistently activated
8	Neuron 8	Clear and focused activity

Summary of Findings

- Best and Worst Network/Input/Training Configurations

  Best Configuration:

  - Input Size: 28×28
  - Time Steps: 10
  - Architecture: Smaller SNN (128–64–10)
  - Training Method: Surrogate Gradient Descent (SGD)
  - Test Accuracy: 97.58%

  Observation: This setup offered the best trade-off between accuracy and computational complexity.

  Worst Configuration:

  - Input Size: 4×4
  - Time Steps: 5
  - Architecture: Any
  - Training Method: SGD
  - Test Accuracy: 26.26%

  Observation: Very limited spatial resolution and minimal temporal encoding led to underfitting.

- Impact of Different Parameters on Accuracy

  Image Size:

  Accuracy increased significantly with larger image sizes.

  Example: Accuracy jumped from 46.61% (4x4, 20 ts) to 95.94% (28x28, 10 ts).

  Time Steps:

  Moderate time steps (10) yielded the best balance between inference latency and spike accumulation.

  Time Step Comparison:

  - 5 ts: Fast but lower accuracy
  - 10 ts: Best balance
  - 20 ts: Slight gain, more latency
  -

Weight Quantization:

  - 3-bit and above preserved accuracy (>95%)
  - 2-bit led to noticeable degradation (90.52%), but still usable

- Most Effective Training Method

  Surrogate Gradient Descent (SGD):

- Achieved high classification accuracy (up to 97.58%)
- Suitable for supervised learning with labeled datasets
- Spike-Timing Dependent Plasticity (STDP):
- Label-free and biologically plausible
- Did not reach competitive classification accuracy
- More suitable for unsupervised feature extraction or pretraining

Conclusion: SGD is more effective for performance-driven classification tasks, while STDP offers advantages for low-power unsupervised neuromorphic applications.

- Additional Observations

  Smaller Network Efficiency:

  - Reduced hidden layer sizes significantly decreased model size and memory needs while still achieving >97% test accuracy
    - This makes it highly suitable for neuromorphic hardware deployment.

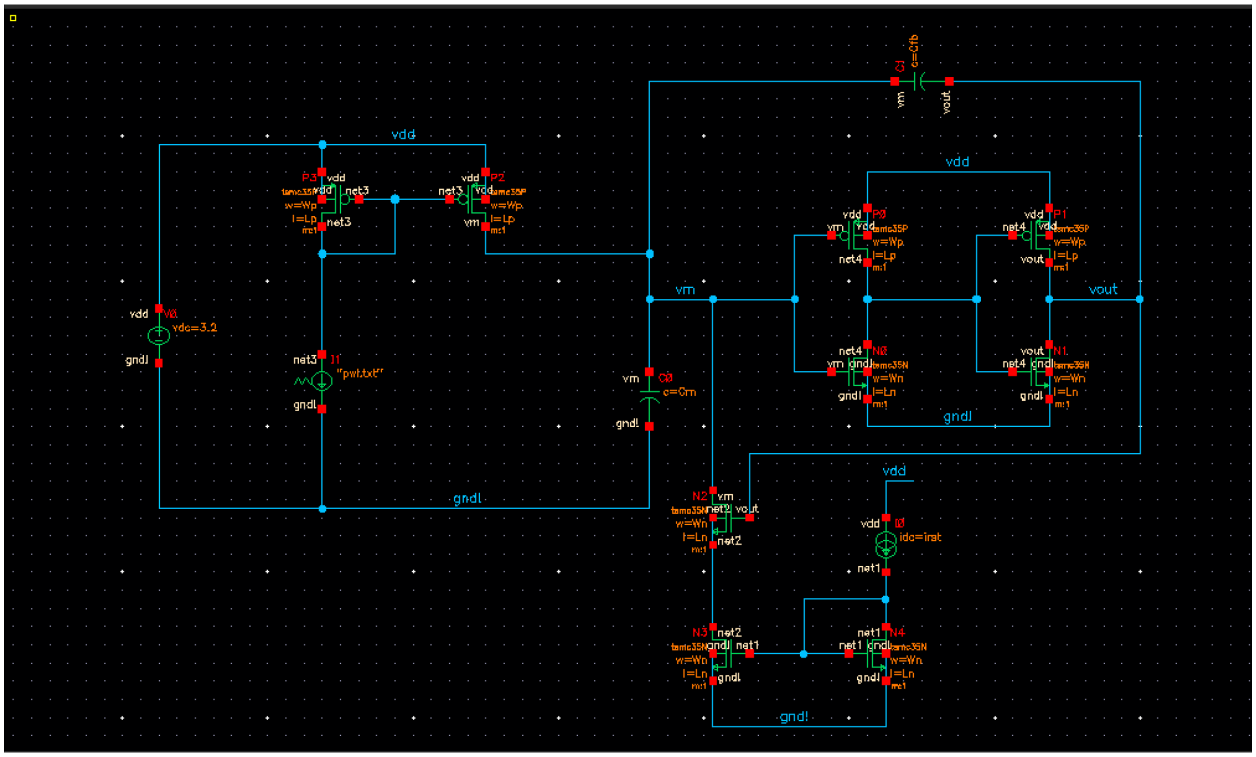
  Spike Raster Plots:

  - Input layer spike activity reflected image structure accurately via Poisson encoding.
  - Hidden layers showed progressively sparser activity.
  - Output layer often had a single neuron spiking consistently for correct class.

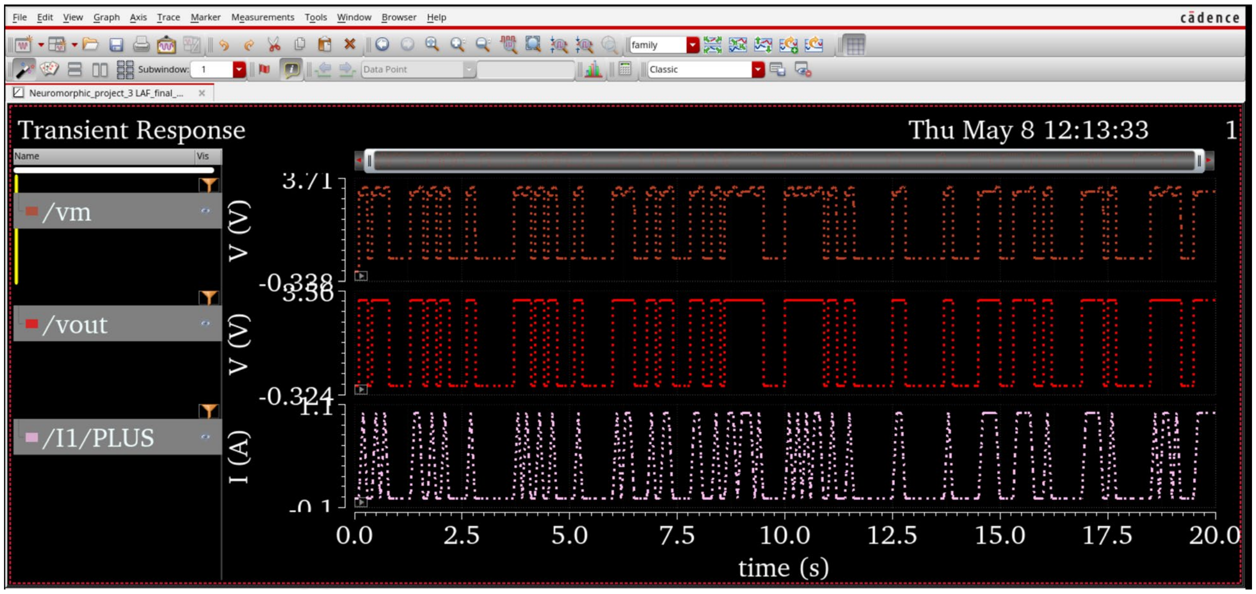
  Synaptic Weights:

  - Histogram showed weights centered around 0 with a Gaussian-like distribution.
  - Indicates stable training and no saturation.

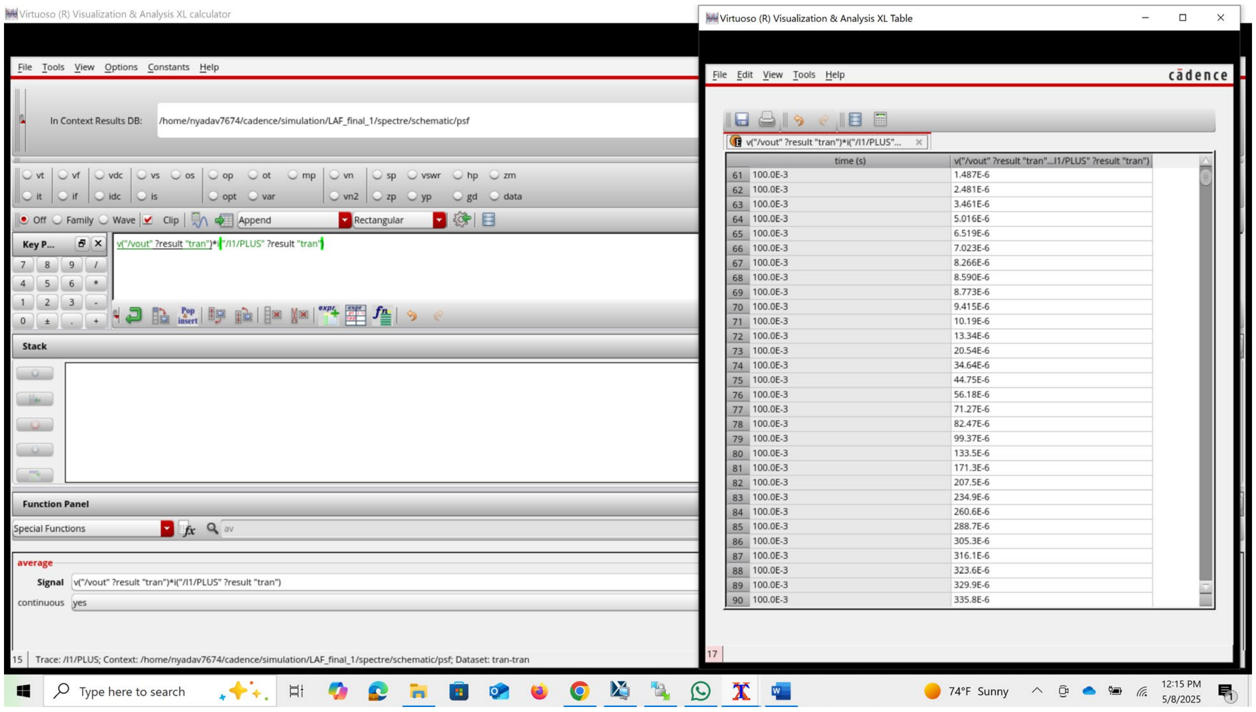
Circuit Snippet –



Cadence Output –



Power output calculation


Virtuoso (R) Visualization & Analysis XL calculator
File Tools View Options Constants Help
Virtuoso (R) Visualization & Analysis XL Table
File Edit View Tools Help
cadence
Key P...
Stack
Function Panel
Special Functions
74°F Sunny
12:15 PM
5/8/2025

In Context Results DB: /home/nyadav7674/cadence/simulation/LAF_final_1/spectre/schematic/psf

Off [✓ Family] Wave [✓ Clip] Append Rectangular

```
v("/vout" ?result "tran")*i("/I1/PLUS" ?result "tran")
```

average
Signal: v("/vout" ?result "tran")*i("/I1/PLUS" ?result "tran")
continuous: yes

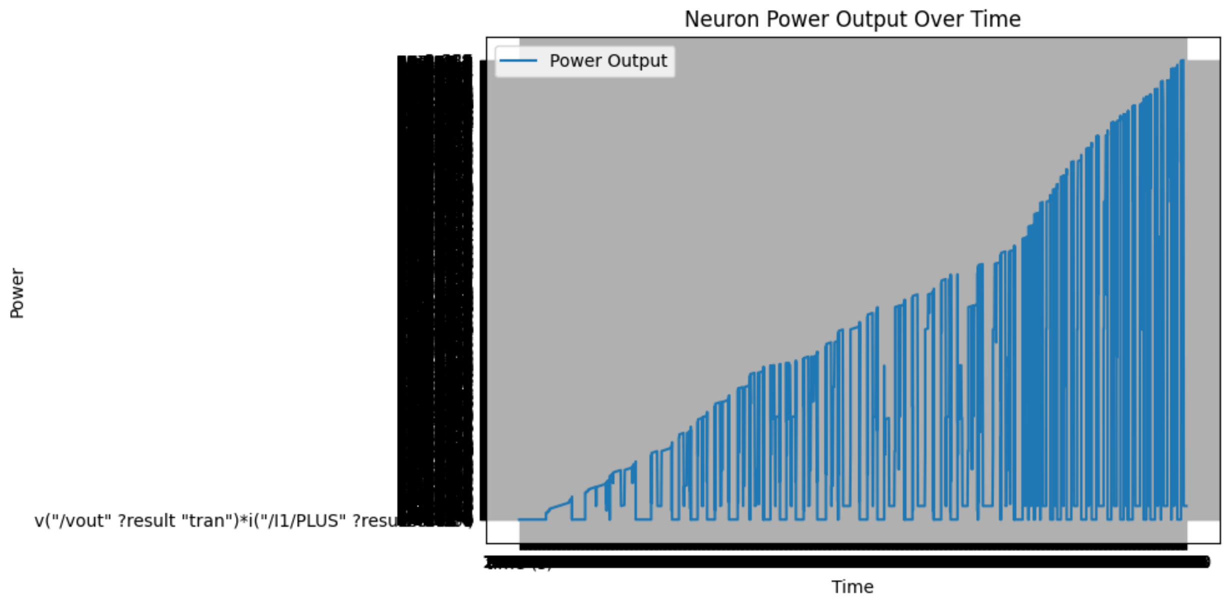
15 | Trace: /I1/PLUS; Context: /home/nyadav7674/cadence/simulation/LAF_final_1/spectre/schematic/psf; Dataset: tran-tran

v("/vout" ?result "tran")*i("/I1/PLUS"...

	time (s)	v("/vout" ?result "tran"...I1/PLUS" ?result "tran")
61	100.0E-3	1.487E-6
62	100.0E-3	2.481E-6
63	100.0E-3	3.461E-6
64	100.0E-3	5.016E-6
65	100.0E-3	6.519E-6
66	100.0E-3	7.023E-6
67	100.0E-3	8.266E-6
68	100.0E-3	8.590E-6
69	100.0E-3	8.773E-6
70	100.0E-3	9.415E-6
71	100.0E-3	10.19E-6
72	100.0E-3	13.34E-6
73	100.0E-3	20.54E-6
74	100.0E-3	34.64E-6
75	100.0E-3	44.75E-6
76	100.0E-3	56.18E-6
77	100.0E-3	71.27E-6
78	100.0E-3	82.47E-6
79	100.0E-3	99.37E-6
80	100.0E-3	133.5E-6
81	100.0E-3	171.3E-6
82	100.0E-3	207.5E-6
83	100.0E-3	234.9E-6
84	100.0E-3	260.6E-6
85	100.0E-3	288.7E-6
86	100.0E-3	305.3E-6
87	100.0E-3	316.1E-6
88	100.0E-3	323.6E-6
89	100.0E-3	329.9E-6
90	100.0E-3	335.8E-6

17

Power output –



Code Files

EE533 - Neuromorphic Final Project.zip

Link - https://drive.google.com/file/d/1tj5wxKzWUm7tFqBeLLyAUtsq5yib1t5i/view?usp=drive_link