Spiking Neural Network Report - MNIST Classification

1. Introduction

This project implements a Spiking Neural Network (SNN) to classify handwritten digits from the MNIST dataset.

SNNs emulate the time-based firing of biological neurons and process data as temporal spike trains.

2. Step-by-Step Approach

- Load MNIST images and convert each image into a spike train using Poisson encoding.
- Model architecture:
    - Input layer: 784 neurons (28x28 image)
    - Hidden layer: 512 LIF neurons
    - Output layer: 10 LIF neurons
- Neuron model: Leaky Integrate-and-Fire (LIF)
    - Leaks over time
    - Fires if threshold is crossed, then resets
- Training:
    - Use surrogate gradients to approximate spike gradients
    - Optimizer: Adam
    - Loss: CrossEntropy between final output spike counts and true labels

3. Visualizations

- Poisson spike trains generated from images show meaningful activity
- Accuracy improves over epochs
- Loss drops significantly within a few epochs

4. Observations & Challenges

- CPU/CUDA mismatch fixed with careful device transfers

- Poisson encoding is stochastic; training curves may vary
- Surrogate gradient selection is crucial
- Higher number of time steps improves accuracy but increases compute

5. SNN vs ANN

Feature	SNN	ANN
Activation	Binary spikes	Continuous values
Training	Surrogate gradients	Backpropagation
Timing	Time-based	Static
Power	Energy efficient (sparse)	Dense computation
Hardware	Neuromorphic-friendly	CPU/GPU

6. Reflections & Future Work

- Accuracy around 92-94% on MNIST is promising
- Explore STDP learning or event-driven datasets like DVS-Gesture
- Investigate deeper SNNs or hybrid SNN-ANN architectures

7. References

- Neftci et al. (2019): Surrogate Gradient Learning in SNNs
- MNIST dataset: http://yann.lecun.com/exdb/mnist/
- PyTorch docs: https://pytorch.org/
- fzenke/spytorch: https://github.com/fzenke/spytorch