ES 673: NeuroAI
Spiking Neural Networks

Question 1 — Conceptual / design (50 marks) - submit as PDF

You are designing a small neuromorphic vision front-end for an always-on camera that should detect a small set of visual events (e.g., “person present”, “no person”, “motion only”). The camera is energy-constrained and should avoid sending raw frames—only event-like/feature information should be transmitted.

1. (10 marks) **Encoding choice:** SNNs use various strategies to encode information into spike trains. The most popular ones are rate encoding and temporal encoding. Compare these spike-encoding strategies for static camera frames: (a) rate-based Poisson encoding (image pixel intensity → spike-rate), and (b) latency/temporal encoding (brighter pixels spike earlier). For each:
    - Describe how encoding is computed.
    - List pros/cons for energy, latency of decision, and robustness to noise.
    - Which encoding would you choose if the system must quickly detect a rare event? Why?

2. (10 marks) **Network architecture:** Propose an SNN architecture (layers, neuron types, connectivity, readout) for the front-end. Be explicit about:
    - Number of layers and their roles.
    - Types of neurons (LIF, adaptive LIF, etc.) and why.
    - Kind of connectivity: sparse random connectivity, or another connectivity motif.
    - How the readout/classifier will be implemented (spiking readout or rate-based classifier after spike accumulation).

3. (10 marks) **Plasticity & adaptation:** Describe what learning rules (eg. STDP) you would deploy and where. Explain:
    - Which rule is used for which synapses and your motivation.
    - How you would adapt the network to new environments after deployment without catastrophic forgetting.

4. (10 marks) **Evaluation & metrics:** Propose an evaluation plan:
    - What datasets or data-collection protocol would you use (synthetic events? real camera data?).
    - Metrics (latency to detection, energy estimate: spikes/sec or average firing rate, accuracy/F1, false alarm rate).
    - Ablation studies you would run (e.g., disable homeostasis, change encoding) and what you expect to observe.

Question 2 — Implementation (50 marks) — Brian2-based SNN experiment

Implement a small feedforward SNN in **Brian2** that receives Poisson spike-encoded MNIST images, uses a hidden layer of LIF neurons, and uses a **linear readout** trained on spike counts. Then implement and test **one** synaptic plasticity/tuning tweak (choice below) and report how it changes network behavior and classification performance.

Baseline model

1. **Input encoding:** Convert each grayscale image (28×28) to Poisson spike trains for a fixed encoding interval T_enc (e.g., 100 ms).
2. **Network architecture:**
    - Input layer: 28×28 Poisson sources (flattened → N_in = 784).
    - Hidden layer: Number of LIF neurons which you determine. Dense or fractional connectivity.
    - Readout: For each image, record spike counts of hidden neurons. Train a linear classifier (sklearn LogisticRegression or RidgeClassifier) to map spike-count vectors → digit label.
3. **Neuron model:** Standard LIF with membrane time constant τ_m, resting potential V_rest, reset V_reset, threshold V_th. Use refractoriness.
4. **Synapses:** You can randomly assign initial weight or can sample them from normal distribution (e.g., N(0.5, 0.1)) clipped to positive for excitatory input.
5. **Training readout:** Freeze SNN weights; train readout offline using spike-count features. Report classification accuracy on test set.

Required: Jupyter Notebook with code implementation, plots and accompanying analysis.

Extra credit

You can modify your baseline architectures with these potential tweaks (choose one) and observe behavior of the model. Does it improve classification accuracy or not?

Homeostatic firing-rate regulation: Neurons optimize for homeostasis unlike artificial neurons which fire based on inputs. Try implementing a target firing rate for each hidden neuron.

Change neuron/time constants: Systematically vary τ_m (e.g., 5 ms, 20 ms, 50 ms) and report how spike timing, spike counts, and classification accuracy change. Provide plots of accuracy vs τ_m and example voltage traces.

Sparse connectivity: Use sparse connectivity (probability of connectivity = 0.05). Compare performance and show a few learned/initial filters.