Classification task on Spiking Heidelberg Digits Dataset

AYUSH BODADE, ayushbodade1@gmail.com

My Code

https://github.com/ayushb03/snn-working-memory-edge-ai (V0: SNN_2, V1: SNN_3, V2: SNN_4)

https://github.com/ayushb03/event-ssm

Research Paper on Deep State-Space Models: Scalable Event-by-event Processing of Neuromorphic Sensory Signals With Deep State-Space Models

Dataset

https://tonic.readthedocs.io/en/latest/generated/tonic.datasets.SHD.html (SHD served by tonic library)

https://zenkelab.org/resources/spiking-heidelberg-datasets-shd/ (raw)

Analysis

https://github.com/ayushb03/snn-working-memory-edge-ai/blob/main/notebooks/SHD_visualisation.ipynb

Data Pre-Processing

The dataset is pre-processed into frames using the SHD2Raster transform, which encodes spikes into binary frames with a time resolution of 128 steps.

```
class SHD2Raster():
    """
    Tool for rastering SHD samples into frames. Packs bits along the tempor
        that the used will have to apply jnp.unpackbits(events, axis=<time
    """

    def __init__(self, encoding_dim, sample_T = 100):
        self.encoding_dim = encoding_dim
        self.sample_T = sample_T

    def __call__(self, events):
```

```
        # tensor has dimensions (time_steps, encoding_dim)
        tensor = np.zeros((events["t"].max()+1, self.encoding_dim), dtype=i
        np.add.at(tensor, (events["t"], events["x"]), 1)
        #return tensor[:self.sample_T,:]
        tensor = tensor[:self.sample_T,:]
        tensor = np.minimum(tensor, 1)
        #tensor = np.packbits(tensor, axis=0) pytorch does not have an unpa
        return tensor
```

Architecture

Input: 700 channels, transformed to 128 time steps.

Model	Network Architecture	Layer Configuration
SNN_2_lif_hidden_64_epochs_100	2 layers of LIF neurons	700 → 64 → 20 (output, no reset)
SNN_3_lif_hidden_64_epochs_100	3 layers of LIF neurons	700 → 64 → 64 → 20 (output, no reset)
SNN_4_lif_hidden_128_epochs_100	4 layers of LIF neurons	700 → 128 → 128 → 128 → 20 (output, no reset)
Deep State-Space Models		

Training

EPOCHS = 100

BATCH SIZE = 256

Model	Loss Function	Optimizer	Learning Rate	Additional Notes
SNN_2_lif_hidden_64_epochs_100	Cross-entropy with **label smoothing** (0.3)	Adam	5e-4	-
SNN_3_lif_hidden_64_epochs_100	Cross-entropy with **label smoothing** (0.3)	Adam	5e-4	Learning rate decay
SNN_4_lif_hidden_128_epochs_100	Cross-entropy with **label smoothing** (0.3)	Adam	5e-4	Step decay every 10 epochs

Deep State-Space Models:

- EPOCHS: 30

- CONFIG: https://github.com/ayushb03/event-ssm/blob/main/outputs/2024-11-08-12-31-56/hydra-outputs/2024-11-08-12-31-56/.hydra/config.yaml

Results

Test Accuracy

SNN_2_lif_hidden_64_epochs_100: **57.23%**

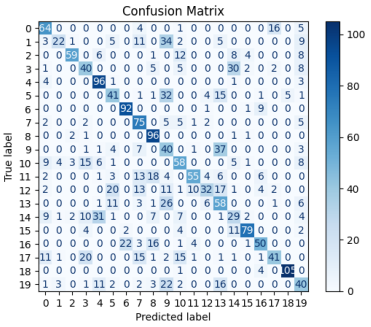
SNN_3_lif_hidden_64_epochs_100: **66.94%**

SNN_4_lif_hidden_128_epochs_100: **71.68%**

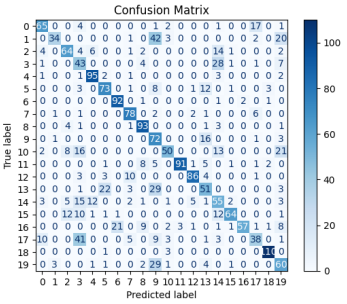
Deep State-Space Model for Neuromorphic Computing: **93.3%**

Confusion Matrix

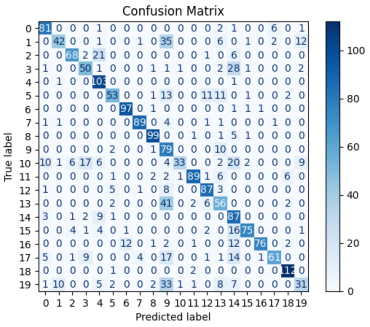
SNN_2_lif_hidden_64



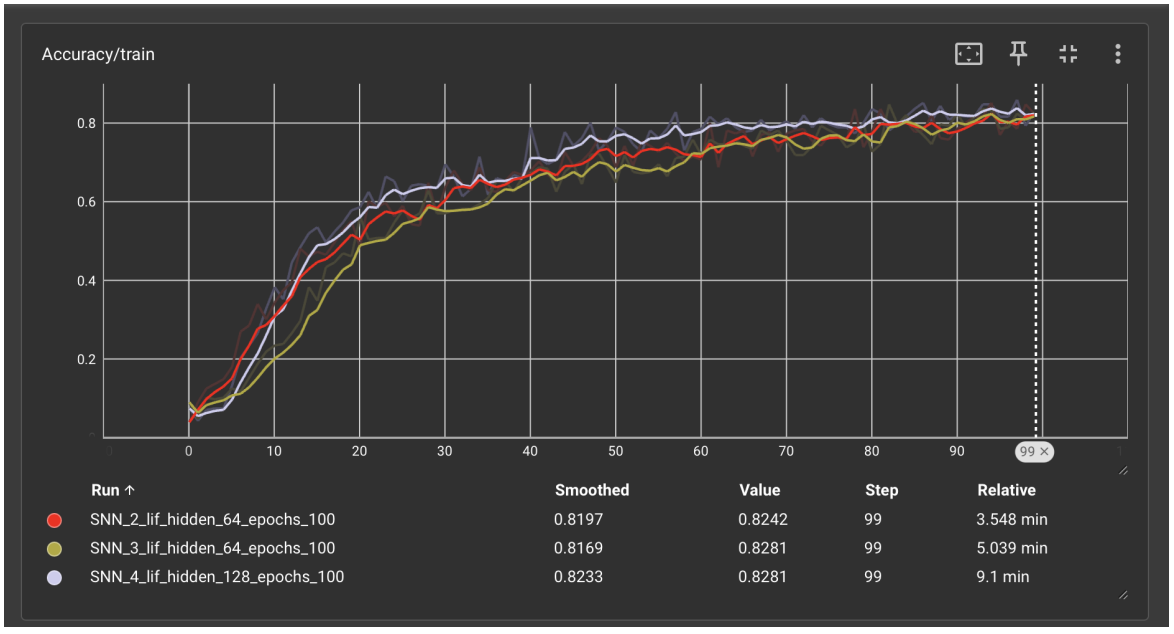
SNN_3_lif_hidden_64



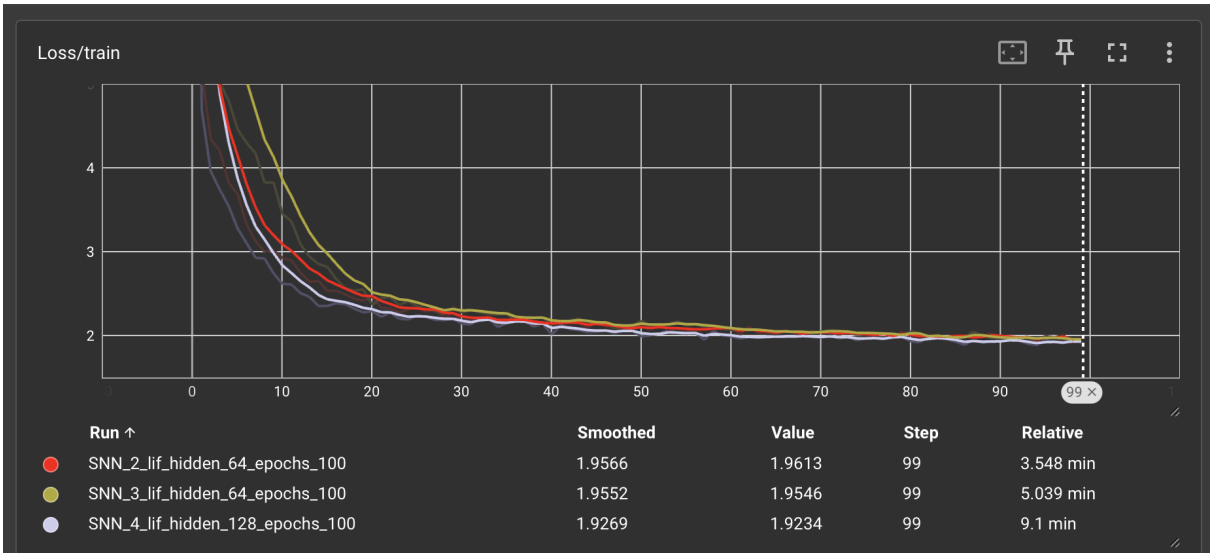
SNN_4_lif_hidden_128



Train Accuracy



Train Loss



Comments

- **SCONV2DLSTM, RLEAKY, SLSTM** show strong potential for improved performance.
- Further learning optimizations can boost accuracy.
- Employing deeper SNNs could enhance results.
- Perhaps Deep State Space Models could be explored