Attention Model using SNN
UCF REU 2022 Week 7

Advisor: Dr. Gonzalo Vaca-Castano
Student: Connor Cabrera
Weekly Overview

- SNN Torch
- Constructed SNN ViT that runs
  - Generating Poor Metrics on MNIST

Difficulties

- Optimization
  - Batch Size / Learning Rate
  - Patch Size
  - Number of Layers / Heads
  - Number of Hidden / MLP Neurons
Corrections from Last Week

Rate Encoding

- There is a spike train per image, not pixel
  - Of Length $W \times H$
- You feed (t) number of spike trains into the model
  - Almost like a video
- My idea of encoding the binary representation defeats any event based nature
  - If intensity creates spikes then 127 (01111111) and 128 (10000000) which are actually similar, would appear different

Heaviside Step Function

- Used by SNN Torch
How Did I Create the SNN ViT

1. Can’t use Pre-Trained Model
   a. Pytorch won’t allow you to transfer weights to a differently structured model (RELU / GELU -> LiF)
   b. Will look into Model Dissection to attempt manual transfer of weights

2. Found two ViT made for MNIST Dataset
   a. Took Structure from one
   b. Took Train / Eval Functions from other

3. Replaced Activation Function with snn.Leaky()
   a. ```python
      self.mlp_lif1 = snn.Leaky(betax = 0.95)
   ```
   b. Forward:
      i. ```python
         mem1 = self.mlp_lif1.init_leaky()
      ```
      ii. ```python
          x, mem1 = self.mlp_lif1(x, mem1)
      ```

4. Rate Encoded Input Data on Training / Evaluation
   a. ```python
      data = spikegen.rate_conv(data)
   ```
Best Results

(So Far...)
No Time
Time Averaged Loss

Average Loss
Train vs Test Accuracy
BATCH SIZE = 256, LR = 0.0003, PATCH SIZE = 7, DIM = 64, DEPTH = 4, HEADS = 8, MLP DIM = 128, TIMESTEPS = 10
Time Averaged Output
What’s Next For Me?

- Convert MNIST dataset to Spike representation and save independently
  - Currently converting with each data pass, would save on runtime
- Increase number of epochs
  - Regular MNIST ViT could converge in less than 25 epochs
  - As long as our model is more efficient in deployment we have achieved our goal
- Add an LiF after every Linear operation instead of just having replaced RELU / GELU
  - Try this on No Time, Average Loss Time, & Average Output Time
- Pre-Trained Model Dissection