SRNN training with differentiable quantization (Stage 2)

Abhijat Bharadwaj, Animesh Kumar, Aditya Kumar Roy (Team ID 6)

November 2023

1 Introduction

Before we begin with the topic of SRNN training with differentiable quantization, we would like to introduce some definitions for the sake of documentation.

1.1 Neural Network

A neural network is a computational model inspired by the structure and function of biological neural networks. It consists of artificial neurons organized into layers. These neurons are connected with associated weights and use activation functions to process data. Neural networks are trained on labeled data to learn patterns and relationships, with deep neural networks having multiple hidden layers. They find applications in various fields, including image and speech recognition, natural language processing, and more.

1.2 Spiking Neural Networks (SNNs)

According to Wikipedia [1], "Spiking neural networks (SNNs) are artificial neural networks that more closely mimic natural neural networks. In addition to neuronal and synaptic state, SNNs incorporate the concept of time into their operating model. The idea is that neurons in the SNN do not transmit information at each propagation cycle (as it happens with typical multi-layer perceptron networks), but rather transmit information only when a membrane potential—an intrinsic quality of the neuron related to its membrane electrical charge—reaches a specific value, called the threshold. When the membrane potential reaches the threshold, the neuron fires, and generates a signal that travels to other neurons which, in turn, increase or decrease their potentials in response to this signal. A neuron model that fires at the moment of threshold crossing is also called a spiking neuron model."

1.3 Recurrent Neural Networks (RNNs)

RNN stands for Recurrent Neural Network. It's a type of neural network designed to process sequential data by maintaining a hidden state that captures information from previous elements in the sequence. RNNs are commonly used in tasks like natural language processing, time series prediction, and speech recognition due to their ability to handle data with temporal dependencies. According to IBM [2], "...They are distinguished by their "memory" as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. While future events would also be helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions."

1.4 Quantization and Its Importance

Quantization is the process of the transformation of continuous, infinite values into a reduced set of discrete, finite values. In our case, quantization is a process of reducing the number of bits to represent synaptic weights. Quantization in neural networks is crucial for addressing the computational and memory limitations that often hinder their deployment on portable and edge computing devices. Deep neural networks, while highly effective in various domains, come with significant computational and memory costs. Edge computing's increasing popularity has created a demand for solutions to overcome these limitations. Quantization offers substantial memory savings and power efficiency by reducing the number of bits used to represent synaptic weights. It is a process that can occur during or

after network training, with quantization-aware training (QAT) and post-training quantization (PTQ) as two main approaches. QAT is often preferred when long-term deployment, hardware efficiency, and accuracy are essential, while PTQ can be a quicker option when there is a need for fast and simple quantization. [3]

2 LSM

Liquid State Machine (LSM) is a neural model with real time computations which transforms the time varying inputs stream to a higher dimensional space. The concept of LSM is a novel field of research in biological inspired computation with most research effort on training the model as well as finding the optimum learning method. The concept of LSM was introduced by Wolfgang Maass and Thomas Natschläger in the early 2000s. The model of LSM comprises of recurrent and randomly spiking neural networks (SNN). One of the main reasons for the preference of SNNs over the second generation artificial neural networks (ANNs) because of its specific property to operate in temporal domain [4]. SNNs are more biologically plausible than ANNs. They are inspired by the way neurons communicate in biological systems, using spikes or action potentials. This biological realism is particularly useful in neuroscience research and neuromorphic computing, where the goal is to mimic the behavior of the human brain more closely. SNNs certainly have some good advantages over ANNs. Some of them are :

- **Temporal Information Processing**
- **Biological Plausibility**
- **Low-power Applications**

Moreover, SNNs are inherently event-driven and operate on the basis of spikes, which are discrete events that occur in response to input stimuli. This event-driven nature can be more power-efficient in scenarios where processing is triggered by specific events rather than continuously.

The key components of a Liquid State Machine include:

- **Input Layer**
- **Reservoir**
- **Readout Layer**

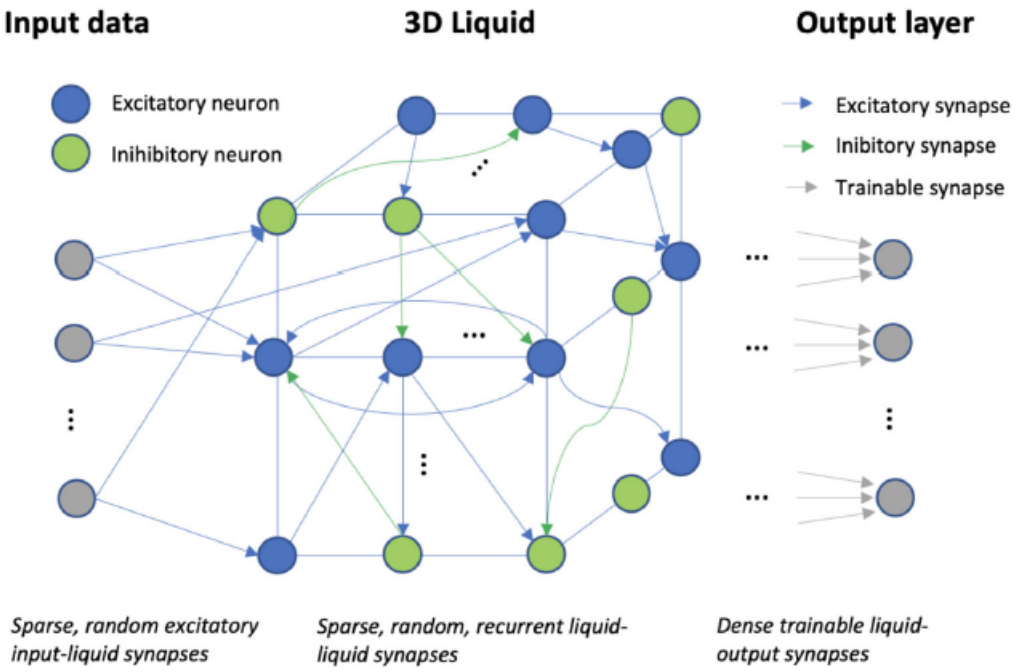

Figure 1: Liquid state Machine (image courtesy: Lucas et al. [5])

3 Qunatization Aware Training

As discussed in Section 1.4 of this document, Quantization Aware Training (QAT) is a training approach that simulates the effects of quantization during the training process. Instead of using full precision (e.g., 32-bit floating-point) for weights and activations, QAT uses lower-precision numerical representations. This helps the model to adapt to the quantization-induced errors during training, leading to a more robust and accurate quantized model.

3.1 Quantization Function

The quantization function presented in [3] is

$$W_Q = \sum_{i=1}^{n} s_i \Theta(\beta W - b_i) - o$$

where $\Theta(x)$ is a unit step function (ie, $= 1 \; if \; x >= 0, \; else = 0$), W is the full-precision weight that needs to be quantized, W_Q is the discrete output value. β is an input scale factor that maps the ranges of W to the range of W_Q. s_i represents the difference between adjacent quantization levels, and b_i defines their border. $o = \frac{1}{2}\sum_{i=0}^{n} s_i$

3.2 Why differentiable quantization?

Since the step function is not differentiable, training feedforward networks directly with the backpropagation method is not feasible. Step functions are replaced with sigmoid functions for training, while step functions are kept for inference. [3]. The sigmoid function we used was:

$$\sigma(Tx) = \frac{1}{1 + e^{-Tx}}$$

Here, the parameter T is called the temperature of the sigmoid function. It controls the gap between two quantization levels. This makes the "differentiable quantization function to be":

$$W_Q = \sum_{i=1}^{n} s_i \sigma(\beta W - b_i) - o$$

Our implementation of the quantization function is as follows:

```
import numpy as np

def sigmoid(T,x):
  return 1/(1+np.exp(-1*T*x))

def unitStep(x):
  val = np.where(x >= 0, 1, 0)
  return val

def quantization_function(
    alpha = 1,
    beta = 1,
    wt=1,
    b=np.arange(-25,25,5),
    s=[1.0]*10, T=0,
    operation = "sigmoid"
    ):
    # s array represents the difference
    # between adjacent quantization levels
    # b array defines their border

    o = 0.5*np.sum(s)
```

```
wq=0

if len(b) != len (s):
    raise Exception(f"length of b and s are different. len-b = {len(b)}. len-s = {len(s)}"
if operation == "unitStep":
    for i in range(len(b)):
      wq+=s[i]*unitStep(beta*wt - b[i])
    wq -= o
elif operation == "sigmoid":
    for i in range(len(b)):
      wq+=s[i]*sigmoid(T, beta*wt - b[i])
    wq -= o
return np.float32(alpha*wq)
```

3.3 Forward Propagation For SNN

The quantization function used for training:

$$W_Q = \alpha \sum_{i=1}^{n} s_i \sigma(\beta W - b_i) - o$$

The quantization function used for inference:

$$W_Q = \alpha \sum_{i=1}^{n} s_i \Theta(\beta W - b_i) - o$$

3.4 Backward pass for SNN

Loss Function :

$$L = \frac{1}{T}\Sigma.\frac{1}{N}\Sigma(S(t,n) - y(t,n))^2$$

where T = timesteps
N = number of classes
S(t,n) = Output spike ; y(t,n) = Actual label at t for class n
Weighted inputs to layer l :

$$X^l(t) = W_Q^{l-1} I^l(t)$$

where W_Q^{l-1} = Quantized Weight
$I^l(t)$ = input spike

4 Observations

Figures 2,3 and 4 show the implementation of a differentiable quantization function (sigmoid quantization) with respect to the non-differentiable (step) quantization function. We can see that as the temperature (T) is slowly increased, the differentiable quantization function begins to coincide with the original quantization function without losing the property of differentiability.

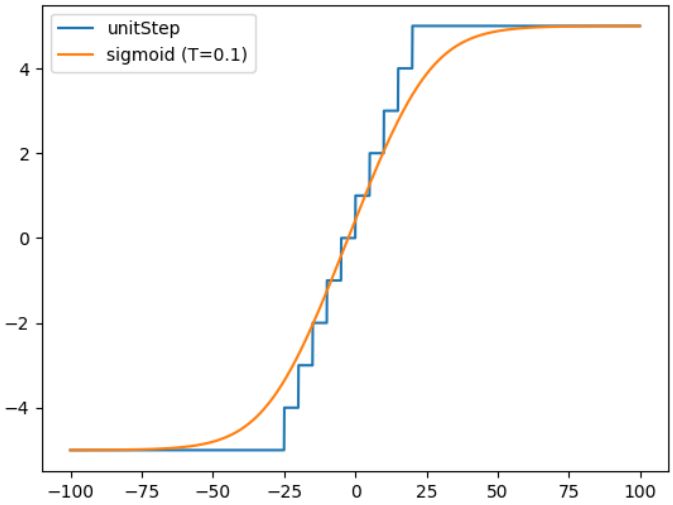

Figure 2: Sigmoid quantization vs Step quantization, T=0.1

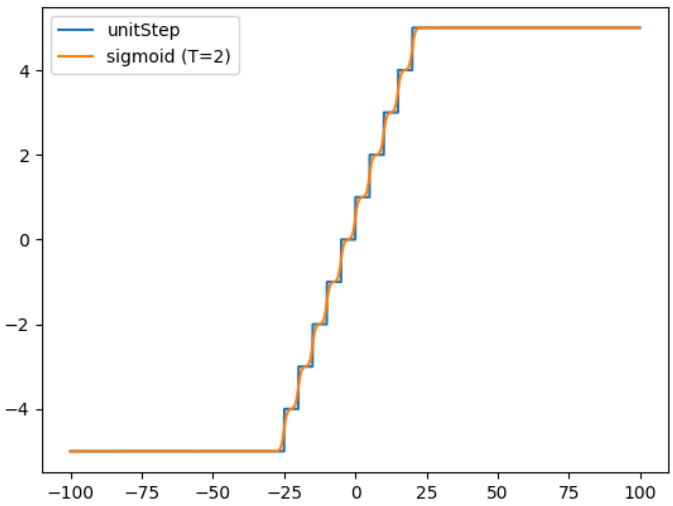

Figure 3: Sigmoid quantization vs Step quantization, T=2

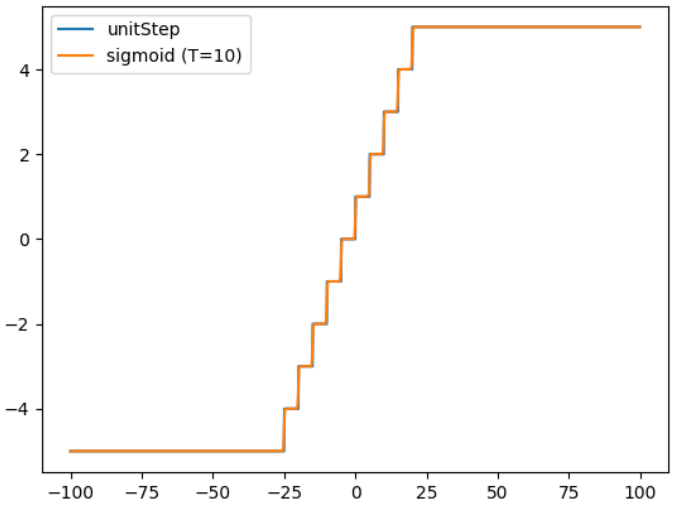

Figure 4: Sigmoid quantization vs Step quantization, T=10

Below (Figure 5) is a 2D scatter plot the Win matrix and its quantized form We also observe that the use of

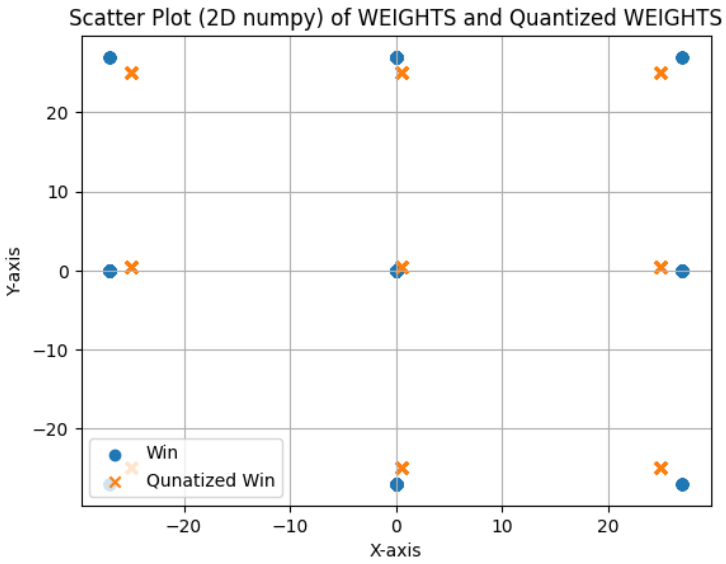

Figure 5: Quantized Win of LSM model

Quantization can cause a significant fall in model training accuracy (see figure 6) Although, there was an unexpected result. While training the SNN with differentiable quantization function over SHD, the loss function was exponentially growing. See Figure 7.

```
(60000, 1000)
(10000, 1000)
(60000, 2312)
(10000, 2312)
mean in spiking (train) :  0.004516701
mean in spiking (test) :  0.0045433217
mean LSM spiking (train) :  0.99746835
mean LSM spiking (test) :  0.9973991
training linear model:
test score = 0.0974
```

Figure 6: Snippet of Test Score of Quantization on LSM-NMNIST

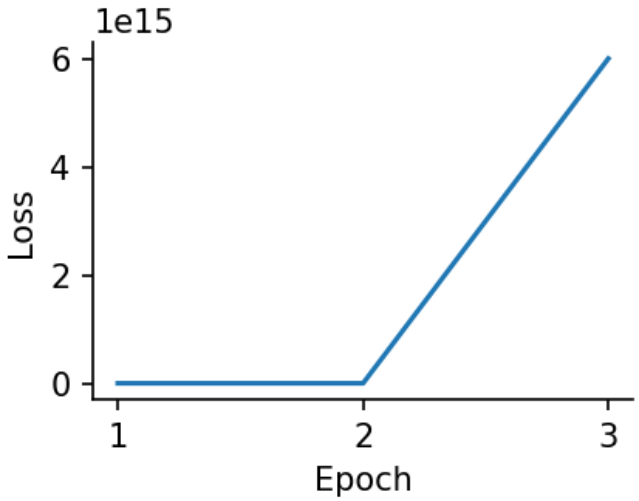

Figure 7: Loss curve of SNN trained over SHD till 3 epochs

5 Inference, Reasoning, and Conclusion

The observations made were mostly in agreement with the literature. Although, our trained models had much lesser accuracy. This could be because of one or many of the below-mentioned reasons:

- The LSM trained was not acquainted by any optimization methods to counter the effects of quantization
- During the training of the NMNIST dataset on the LSM model, the temperature of the quantization function was increased slowly over multiple batches instead of multiple epochs. This decision was made due to long training durations.
- In the SNN trained over SHD dataset, the implementation of the step function could be incorrect in the custom Adam optimizer. This would also explain the exponentially growing loss. (Figure 7)

References

[1] Wikipedia. *Spiking neural network*. [Online; accessed 2023-08-11]. URL: https://en.wikipedia.org/wiki/Spiking_neural_network.

[2] IBM. *What are recurrent neural networks?* [Online; accessed 2023-08-11]. URL: https://www.ibm.com/topics/recurrent-neural-networks.

[3] Ayan Shymyrbay, Mohammed E. Fouda, and Ahmed Eltawil. "Low Precision Quantization-aware Training in Spiking Neural Networks with Differentiable Quantization Function". In: v1 (2023), pp. 1, 2, 3. DOI: https://doi.org/10.48550/arXiv.2305.19295. URL: https://arxiv.org/abs/2305.19295.

[4] Gideon, Gbenga, and Oladipupo. "Low Precision Quantization-aware Training in Spiking Neural Networks with Differentiable Quantization Function". In: v1 (2019), p. 1. DOI: https://doi.org/10.48550/arXiv.1910.03354. URL: https://arxiv.org/abs/1910.03354.

[5] Lucas Deckers et al. "Extended liquid state machines for speech recognition". In: v16 (2022), pp. 5, 9. DOI: https://doi.org/10.3389/fnins.2022.1023470. URL: https://www.frontiersin.org/articles/10.3389/fnins.2022.1023470/full.