SRNN Training With Differentiable Quantization

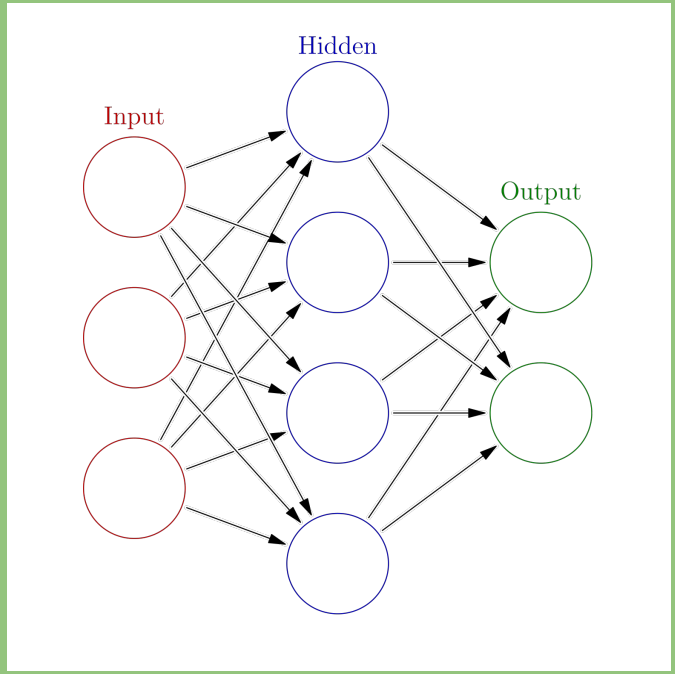
By Abhijat, Animesh and Aditya
(210020002, 21D070012, 21D070008)
Project ID: 15 Team ID: 06

Neural Networks

Artificial Neural Networks : ANNs typically are a computational model inspired by the human brain. An artificial neural network is composed of interconnected nodes, or artificial neurons, organized into layers. The layers typically include an input layer, one or more hidden layers, and an output layer.

The neural network processes information through the connections between neurons, where each connection has an associated weight. During training, the network adjusts these weights based on input data and target outputs to learn patterns and make predictions or classifications.

Spiking Neural Networks : SNNs is a type of artificial neural network that is inspired by the functioning of biological neural networks, particularly the spiking behavior of neurons in the brain



Why SNN over ANN ?

- **Temporal processing**: Specialized for modelling the **temporal dynamics** of neural information processing. Use of spikes allows SNNs to capture the precise timing of events, making them suitable for applications where the order and timing of inputs matter.
- **Energy Efficiency** : Consumes power only at the time of spikes thus improving the efficiency compared to continuous computation in ANNs.
- **Cognitive Computing** : SNNs are explored in the context of cognitive computing, where the emphasis is on building systems that can learn, reason, and make decisions in a manner similar to human cognition.

Now, Let's take a closer look on the **Differentiable Quantization Aware Training**

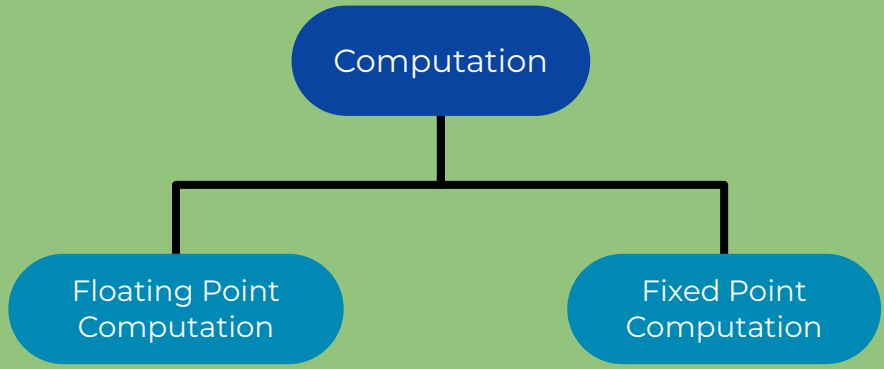
Why This?

Floating Point Computation :

- Computation on variable bits
- High precision
- wide dynamic range
- Complex hardware implementation
- Computationally expensive

Fixed Point Computation :

- Uses a consistent predetermined no. of bits
- Predictable and Efficient hardware
- Optimized Power Efficiency
- Less memory footprint
- increased computational speed.



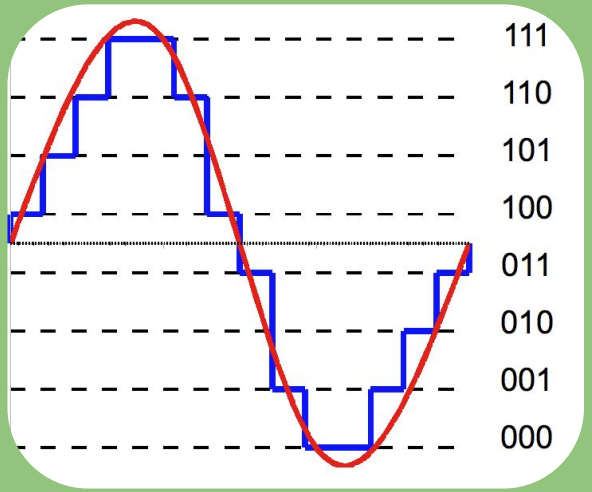
Many hardware accelerators, like those found in edge devices e.g, IoT devices and mobile support fixed point arithmetic more efficiently than floating point computation.

Quantization Aware Training

Quantization-Aware Training (QAT) is needed for fixed-point computation to ensure that a neural network model can effectively operate with reduced precision during both training and deployment.

Key Reasons to perform QAT :

1. Robustness : By training the model with awareness of the quantization process, the model learns to handle the loss of precision that occurs when moving from higher precision (e.g., 32-bit floating point) to lower precision (e.g., 8-bit fixed point).
2. Hardware Adaptation : QAT allows the model to be fine-tuned for specific hardware characteristics, including the precision and format of fixed-point representations supported by the target deployment platform.



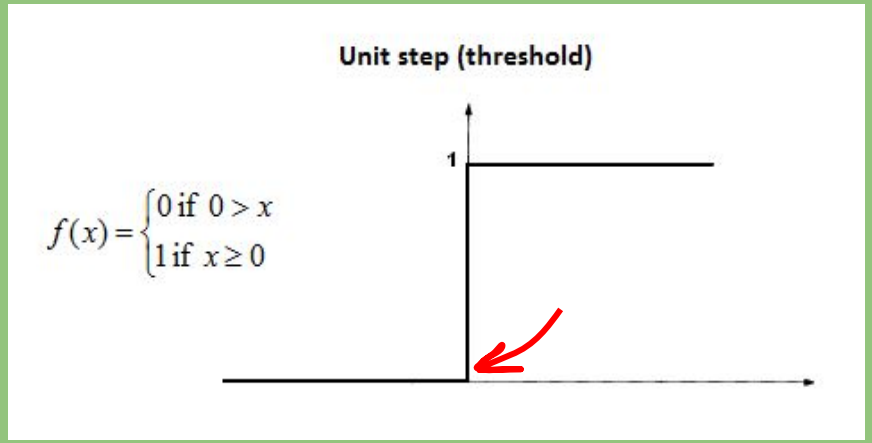
Challenges to QAT

Major problem is the ***Non Differentiability***. When quantizing weights and activations, the quantization operation becomes non-differentiable as it introduces step-like functions, making it challenging to perform gradient-based optimization during backpropagation. In the presence of non-differentiable operations, optimization algorithms like traditional gradient descent methods cannot be directly applied.

Other challenges could be

- Loss of information during training
- Hardware compatibility

In this presentation, we are going to address the problem of non-differentiability and how to mitigate it.



Approach to QAT

- Based on linear combination of Sigmoid function (σ)

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

Where W = full-precision weight

W_Q = discrete output value,

Θ = unit step function

β = input scale factor

S_i = diff. between quantization levels,

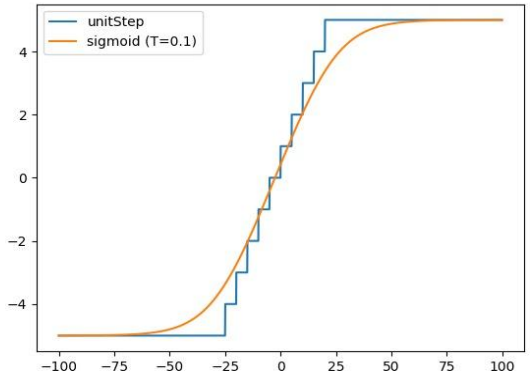
b_i = border of levels

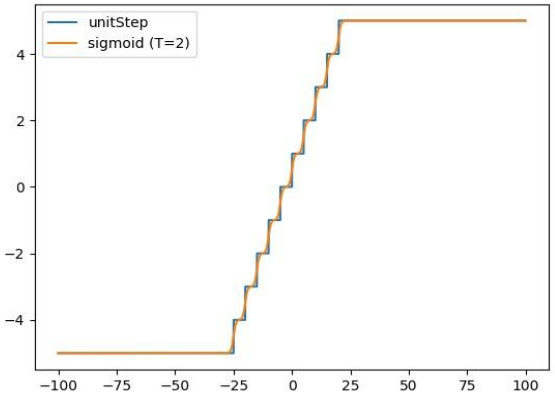
$$o = \frac{1}{2} \sum_{i=1}^{n} s_i$$

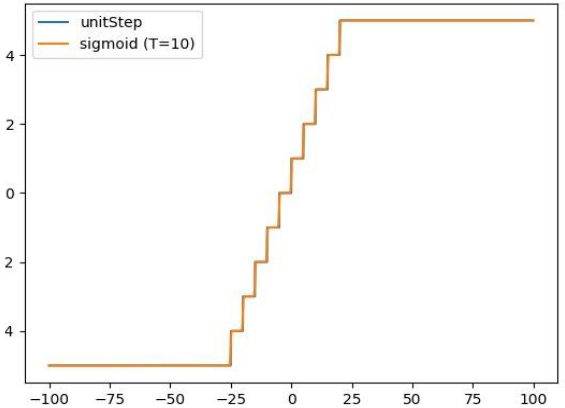
Θ : replaced by a sigmoid function for making it differentiable

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

Effect of increasing T :







Forward Pass

Inference stage:

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

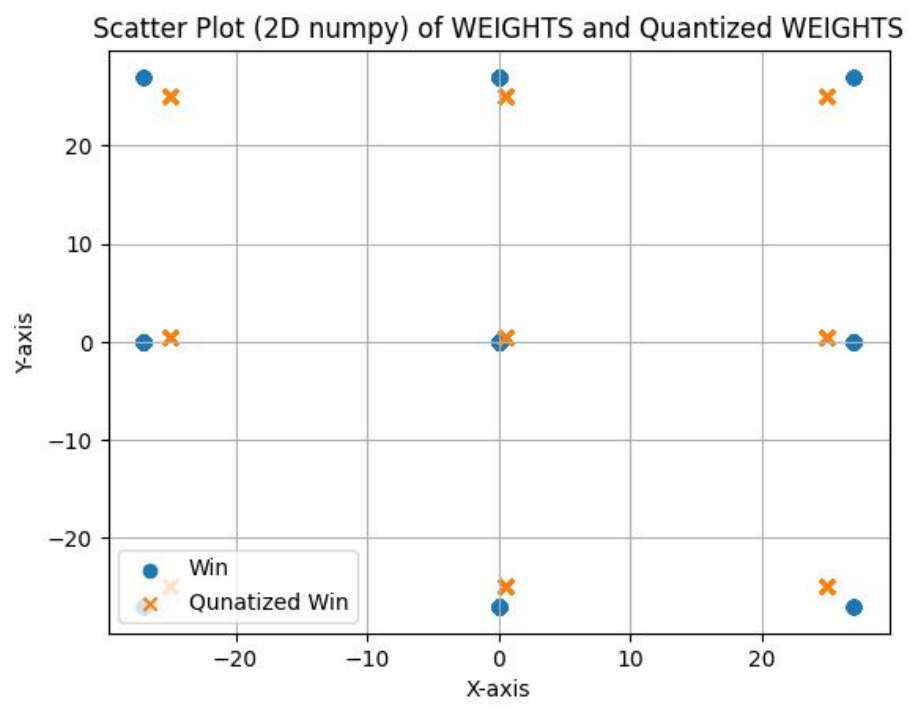
Training Stage :

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

- Weights of each layers : mapped to discrete values
- Inference Stage : Step Function is used for quantization
- Training Stage : Sigmoid Function is used for quantization
- Parameters : *α* and *β*, scale factors
- HyperParameters : b_i , *T,* s_i , *o.*

Observations

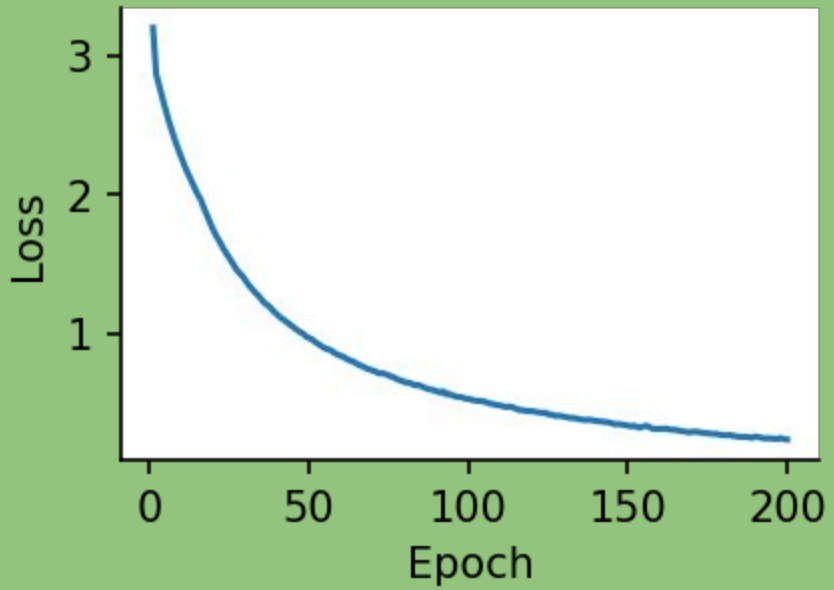
Effect Of Quantization



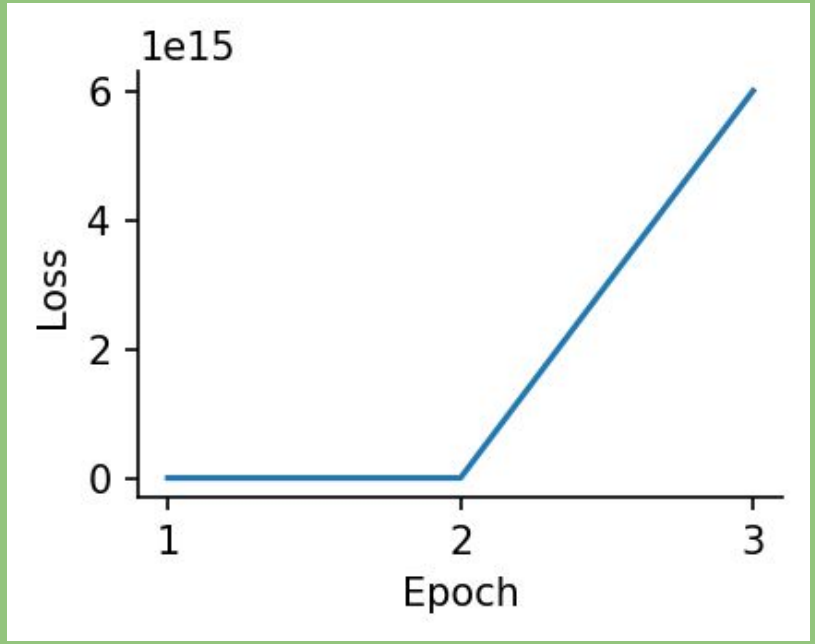
Loss In Accuracy

```
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
```

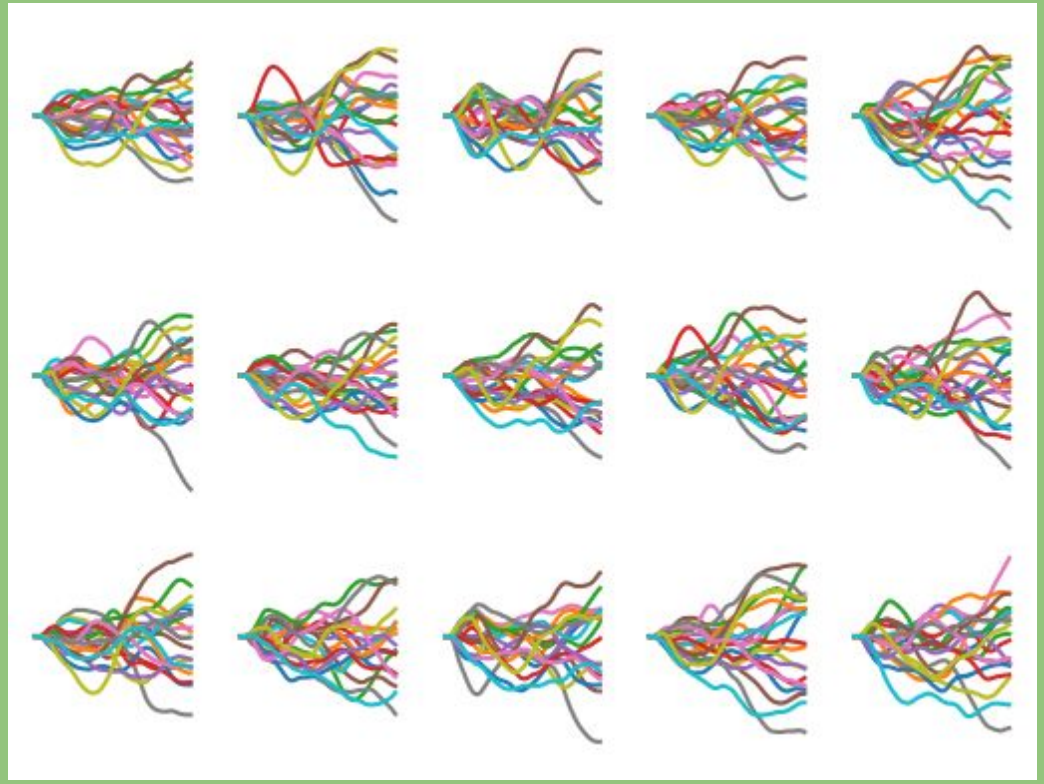
Loss Curve On training SNN For SHD Dataset



Anomaly Found



Anomaly Found



Inference and Conclusion