WSAI IITM

Internship Report

ANN-to-SNN Conversion with QCFS and Testing Effect of Regularizer

Submitted by

Naman Kaur Gill

Under the guidance of

Dr. Gopalakrishnan Srinivasan

Indian Institute of Technology Madras

May – July 2025

Acknowledgement

I would like to express my sincere gratitude to Dr. Gopalakrishnan Srinivasan for his invaluable guidance and support throughout my internship. His expert supervision and thoughtful feedback played a crucial role in shaping the direction and quality of this work.

I am also thankful to the Wadhwani School of Data Science and Artificial Intelligence (WSAI) for providing me with this opportunity. The intellectually enriching environment and access to academic resources significantly contributed to my technical and professional development.

Lastly, I am deeply grateful to my fellow interns for their support and helpful discussions, my friends and family for their constant encouragement, understanding, and unwavering support throughout this journey.

Contents

Chapter 1

Introduction

Spiking Neural Networks (SNNs) mimic brain neurons and serve as an alternative to standard neural networks. Unlike Artificial Neural Networks (ANNs), which operate on continuous values and require extensive computation, SNNs rely on sparse, binary spike signals emitted only when a neuron's membrane potential crosses a threshold. This spike-based processing offers significant advantages like low power consumption and fast inference on neuromorphic hardware.

Training deep SNNs is still a major challenge, primarily due to the non-differentiable nature of spikes and the added complexity of time-based computations. To train SNNs, two main techniques are used: one uses surrogate gradients to make back-propagation feasible in SNNs, and the other converts a pre-trained ANN into an SNN by substituting ReLU activations with spiking neuron models like the Integrate-and-Fire (IF) neuron. The latter method, known as ANN-to-SNN conversion, is used more widely because it takes advantage of existing ANN training techniques and leads to better scalability.

There are some issues with the conversion approach. One of the key issues lies in the accuracy drop that occurs when the number of time-steps decreases. Three types of errors are identified while converting, namely clipping error, quantization error, and unevenness error. Although recent work has made progress in reducing clipping and quantization errors, unevenness error remains stubborn. Unevenness error arises from how spikes accumulate unevenly across layers and time-steps, often disrupting the expected behavior of the converted network.

During my internship, I explored whether unevenness error can be reduced through the use of regularization during ANN training. The project focused on employing the Quantization Clip–Floor–Shift (QCFS) activation function, which was designed to make the ANN outputs more compatible with SNN behavior. Alongside this, I experimented with regularizers aimed at enforcing smoother activations that would translate into more stable spike patterns after conversion. The broader goal was to improve the reliability and performance of SNNs under low latency, where only a few time-steps are allowed.

Chapter 2

Background

2.1 Neuron Models

2.1.1 ANN Neuron Model

In an Artificial Neural Network (ANN), each neuron performs a weighted sum of inputs followed by a non-linear activation function. For a neuron in layer l, the output activation is:

$$a^l = h(W^l a^{l-1} + b^l), \quad h(x) = \max(0, x)$$

Here:

1. W^l is the weight matrix for layer l,
2. b^l is the bias vector,
3. $h(x)$ is typically the ReLU activation function.

2.1.2 SNN Neuron Model (Integrate-and-Fire)

In Spiking Neural Networks (SNNs), the neuron integrates inputs over time and emits a spike when the membrane potential crosses a threshold. For a neuron in layer l at time t, the dynamics are:

$$m^l(t) = v^l(t-1) + W^l x^{l-1}(t)$$

$$s^l(t) = H(m^l(t) - \theta^l), \quad v^l(t) = m^l(t) - \theta^l s^l(t)$$

Where:

1. $m^l(t)$ is the membrane potential before firing,
2. $v^l(t)$ is the updated membrane potential,
3. $x^{l-1}(t)$ is the spike input from the previous layer at time t,
4. θ^l is the firing threshold,
5. $H(\cdot)$ is the Heaviside step function:

$$H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

2.2 ANN-to-SNN Conversion

The fundamental objective of ANN-to-SNN conversion is to establish a correspondence between the activation value of an analog neuron in an Artificial Neural Network (ANN) and the firing rate or average postsynaptic potential of a spiking neuron in a Spiking Neural Network (SNN). By combining Equations (2)–(4), the membrane potential update rule for spiking neurons during conversion is given by:

$$v^l(t) - v^l(t-1) = W^l x^{l-1}(t) - s^l(t)\theta^l \tag{2.1}$$

Equation (6) encapsulates the core membrane potential dynamics in the converted SNN. To analyze this behavior over a time window, we sum both sides from $t = 1$ to T and divide by T, yielding:

$$\frac{v^l(T) - v^l(0)}{T} = W^l \left(\frac{1}{T} \sum_{i=1}^{T} x^{l-1}(i) \right) - \left(\frac{1}{T} \sum_{i=1}^{T} s^l(i)\theta^l \right) \tag{2.2}$$

Let $\phi^{l-1}(T) = \frac{1}{T} \sum_{i=1}^{T} x^{l-1}(i)$ represent the average postsynaptic potential over the interval $[0, T]$. Substituting this definition and Equation (5) into Equation (7), we obtain:

$$\phi^l(T) = W^l \phi^{l-1}(T) - \frac{v^l(T) - v^l(0)}{T} \tag{2.3}$$

Equation (8) characterizes the relationship between the average postsynaptic potentials of adjacent layers in the SNN. It is important to note that $\phi^l(T) \geq 0$. If the initial membrane potential is assumed to be zero, i.e., $v^l(0) = 0$, and the residual term $\frac{v^l(T)}{T}$ is negligible for sufficiently large T, then the converted SNN approximates the behavior of the source ANN defined in Equation (1). However, a large time window T increases inference latency, thereby hindering the deployment of SNNs in latency-sensitive applications. Consequently, this work seeks to achieve high-accuracy ANN-to-SNN conversion under extremely low time-step constraints.

2.3 Conversion Error Analysis

This section presents a detailed analysis of the conversion error introduced between the source ANN and the converted SNN on a per-layer basis. We assume that both the ANN and SNN receive an identical input from the preceding layer, such that $a^{l-1} = \phi^{l-1}(T)$. In this context, the weighted input to layer l for both networks is denoted by:

$$z^l = W^l \phi^{l-1}(T) = W^l a^{l-1}$$

The absolute conversion error in layer l is defined as the difference between the output of the converted SNN and the activation of the source ANN:

$$\mathrm{Err}^l = \phi^l(T) - a^l = z^l - \frac{v^l(T) - v^l(0)}{T} - h(z^l) \tag{2.4}$$

Here, $h(z^l)$ denotes the ReLU activation function applied in the ANN. From Equation (9), it is evident that the conversion error becomes non-zero if $v^l(T) - v^l(0) \neq 0$ and $z^l > 0$. This discrepancy arises due to the temporal integration behavior of spiking neurons, which does not perfectly emulate the instantaneous activations in ANNs. In the subsequent section, we categorize the sources of this conversion error and discuss potential mitigation strategies.

2.4 Conversion Errors

When converting Artificial Neural Networks (ANNs) into Spiking Neural Networks (SNNs), discrepancies arise due to differences in how activations are represented. These discrepancies are generally classified into the following three types of errors:

1. Clipping Error

Clipping error occurs when the output activations of the ANN exceed the representable range of the SNN. While ANN outputs can take on large continuous values, SNN outputs are typically limited by the firing threshold θ^l, and spike counts are bounded by the number of time-steps T. To fit within this constraint, large ANN outputs are *clipped* to the maximum possible spike-based value. This causes a loss of information, particularly for high-activation neurons.

Formally, if the ANN output a^l exceeds θ^l, it is clipped as:

$$\phi^l(T) = \text{clip}\left(\frac{\theta^l}{T}\left\lfloor\frac{a^l T}{\lambda^l}\right\rfloor, 0, \theta^l\right),$$

where λ^l is a scaling factor used to normalize a^l before conversion.

2. Quantization Error (Flooring Error)

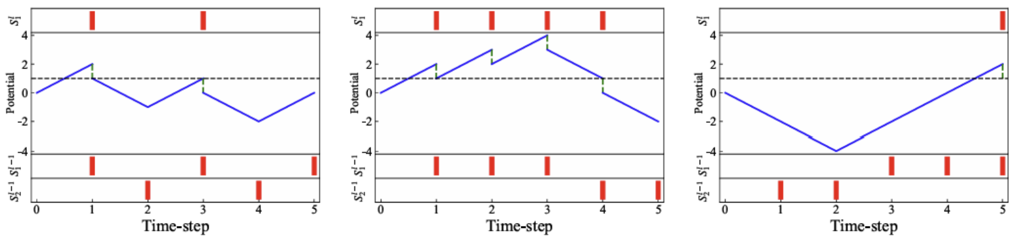
Quantization error arises due to the *discretization* of continuous ANN outputs into a finite number of spikes in SNNs. Since SNN neurons can only emit an integer number of spikes over a finite time window, real-valued ANN activations must be rounded down (via the floor function). This introduces a difference between the original and converted values.

For example, an ANN activation of 0.83 with 4 time-steps might correspond to $0.83 \times 4 = 3.32$ spikes, which is floored to 3 in the SNN, causing a quantization loss of 0.32.

3. Unevenness Error

Unevenness error is caused by the *temporal distribution* of input spikes. Even if the total number of spikes is correct, their timing can vary, which affects how they are integrated by the SNN neuron over time. This error becomes critical when spikes are not uniformly distributed across time-steps, leading to unexpected membrane potential dynamics and ultimately, different outputs compared to the original ANN.

For instance, if two input neurons generate the same number of spikes but one fires early and the other late, the timing mismatch can result in a different firing pattern in the postsynaptic neuron. This makes the SNN output diverge from the ANN even if total spike counts are preserved.



(b) Even spikes (c) More spikes (d) Fewer spikes

Figure 2.1: Illustration of spike timing patterns and their effect on the membrane potential of a postsynaptic neuron. The red bars indicate the spike timings of two presynaptic neurons in layer $l-1$, and the blue curve shows the evolution of the membrane potential of the postsynaptic neuron in layer l. (b) Balanced spikes lead to stable potential oscillations. (c) Increased spike rate causes premature threshold crossing. (d) Reduced spikes delay or prevent firing.

2.5 Quantization Clip–Floor Activation Function

To minimize the conversion error between ANN and SNN outputs, the commonly used ReLU activation function can be replaced with a quantized clip–floor function. This function discretizes activations during ANN training to better align them with the discrete nature of SNN outputs, particularly under low-latency conditions.

The quantized activation function is defined as:

$$a^l = \hat{h}(z^l) = \lambda^l \cdot \text{clip}\left(\frac{1}{L}\left\lfloor \frac{z^l L}{\lambda^l} \right\rfloor, 0, 1\right),$$

where λ^l denotes the upper bound of the activation range in layer l, and L is the quantization level. The floor operation enforces discretization, while the clip function bounds the output within the interval $[0, \lambda^l]$.

Under specific conditions, this formulation leads to zero conversion error. These conditions are outlined in the following theorem:

Theorem 1. *An ANN with activation function as defined above, when converted to an SNN with the same weights and with $T = L$, $\theta^l = \lambda^l$, and $v^l(0) = 0$, results in zero estimated conversion error:*

$$\widetilde{Err}^l = \phi^l(T) - a^l = 0.$$

This result implies that when the number of SNN time-steps matches the ANN quantization steps, and thresholds are properly aligned, the conversion error can be completely eliminated. However, in practical scenarios, the number of time-steps T in the SNN and the quantization resolution L in the ANN may differ. This mismatch can result in nonzero conversion error, which may propagate across layers and affect the overall accuracy of the network.

2.5.1 Quantization Clip–Floor–Shift Activation Function

To address the limitation of mismatched quantization and time-steps, a shift parameter φ is introduced to the activation function. This extension enhances the flexibility of the quantized output and reduces the conversion error even when $T \neq L$.

The modified activation function is expressed as:

$$a^l = \hat{h}(z^l) = \lambda^l \cdot \mathrm{clip}\left(\frac{1}{L} \left\lfloor \frac{z^l L}{\lambda^l} + \varphi \right\rfloor, 0, 1 \right).$$

Here, φ serves as a shift vector that offsets the quantization bins to better match the expected spike counts in SNNs. When $\varphi = 0.5$, the quantization process is centered, and the expected conversion error can be minimized over a range of time-steps.

This is formally stated as follows:

Theorem 2. *An ANN with the above activation function, converted to an SNN with the same weights, and satisfying $\theta^l = \lambda^l$, $v^l(0) = \theta^l \varphi$, yields zero expected conversion error for arbitrary T and L when $\varphi = \frac{1}{2}$:*

$$\forall T, L, \quad \mathbb{E}_{z^l}\left[\widetilde{Err}^l \right] \Big|_{\varphi=\frac{1}{2}} = 0.$$

This result demonstrates that the shift-enhanced clip–floor function is effective in reducing expected conversion error in practical settings where exact matching of time-steps and quantization levels is not guaranteed. It enables reliable ANN-to-SNN conversion across a wider range of system constraints without additional computational cost.

Chapter 3

Simulations

Carried out the simulations to evaluate the performance of ANN-to-SNN conversion using Quantization Clip–Floor–Shift (QCFS) activation. Two well-known architectures, VGG-16 and ResNet-20, were trained and converted on the CIFAR-10 dataset. The experiments investigate the impact of varying quantization steps L and inference time-steps T on classification accuracy.

VGG-16 on CIFAR-10

The following table reports the test accuracy (%) of the converted Spiking Neural Network (SNN) for different values of L and T, using the VGG-16 architecture on CIFAR-10:

$L \backslash T$	4	8	16	32	64	128
4	93.58%	94.82%	95.20%	95.36%	95.33%	95.31%
8	93.24%	95.00%	95.56%	95.75%	95.65%	95.67%
16	92.00%	94.56%	95.49%	95.62%	95.79%	95.84%
32	90.28%	94.08%	95.49%	95.89%	95.93%	95.87%

Table 3.1: Test accuracy (%) of converted SNN for varying quantization steps L and time-steps T using VGG-16 on CIFAR-10.

A consistent improvement in accuracy is observed with increasing quantization levels L and time-steps T. The accuracy stabilizes beyond $T = 32$, indicating that larger time windows provide diminishing returns once temporal resolution is sufficiently high.

ResNet-20 on CIFAR-10

The ResNet-20 architecture was similarly evaluated using the same parameter settings. The results are presented in the following table:

$L\backslash T$	4	8	16	32	64	128
4	83.71%	90.01%	92.11%	92.66%	92.65%	92.67%
8	76.75%	88.12%	92.30%	93.15%	93.15%	93.23%
16	61.06%	81.37%	91.02%	92.95%	93.49%	93.52%
32	35.62%	65.96%	88.00%	92.92%	93.34%	93.05%

Table 3.2: Test accuracy (%) of converted SNN for varying L and T using ResNet-20 on CIFAR-10.

The results show a significant improvement in classification accuracy as the number of time-steps increases, particularly for larger values of L. This trend highlights the importance of temporal resolution in deeper networks like ResNet, where residual connections and nonlinear transformations require more accurate spike-based approximations for effective ANN-to-SNN conversion.

Chapter 4

QCFS Regularization

4.1 Introduction to Regularization

Regularization is a technique used in machine learning to impose additional constraints on the learning process, often to prevent overfitting and encourage generalization. In neural networks, regularizers can also shape the distribution of activations or weights to meet desired properties, such as sparsity, binarization, or quantization alignment.

4.2 Quantization Clip-Floor-Shift (QCFS) Regularizer

In the context of ANN-to-SNN conversion, the Quantization Clip-Floor-Shift (QCFS) activation is employed to align ANN activations with discrete spike counts in the converted SNN. To further encourage activations to lie close to quantized levels, a corresponding regularization term is introduced.

The QCFS activation is defined as:

$$\mathrm{QCFS}(z) = \lambda \cdot \left(\left\lfloor \frac{zL}{\lambda} + 0.5 \right\rfloor - 0.5 \right) \cdot \frac{1}{L}$$

where λ is the maximum activation range, L is the quantization resolution, and $\lfloor \cdot \rfloor$ denotes the floor function.

The associated regularization loss is defined as:

$$\mathcal{L}_{\text{QCFS}} = \frac{1}{N}\sum_{i=1}^{N}(z_i - q_i)^2, \quad \text{where } q_i = \text{QCFS}(z_i)$$

Here, z_i represents the pre-activation output, and q_i is its quantized counterpart obtained via QCFS.

4.3 Mechanism of QCFS Regularizer

The QCFS regularizer enforces alignment of activations with discrete spike-compatible levels through the following steps:

1. **Clipping:** The activation values z are clipped to the range $[0, \lambda]$.
2. **Flooring with shift:** The shifted flooring operation with offset $+0.5$ is used to implement nearest-bin rounding.
3. **Scaling and centering:** The result is scaled back by $1/L$ and offset by -0.5, ensuring zero-mean rounding error.
4. **Regularization loss:** The term $(z-q)^2$ penalizes deviation from the nearest quantized level, effectively pulling activations toward the bin centers.

4.4 Experimental Setup

To evaluate the impact of the QCFS regularization weight w_q on model accuracy, experiments were conducted on VGG-16 using the CIFAR-10 dataset. The quantization level was fixed at $L = 8$, and the number of inference time-steps T varied across the set $\{1, 2, 4, 8, 16, 32, 64, 128\}$. The regularization weight w_q was varied across $\{0.0, 0.1, 0.2\}$.

4.5 Observation

The application of QCFS regularization improves model performance, particularly under low latency conditions. As the regularization weight w_q increases, accuracy improves significantly for smaller time-steps $T \leq 8$, indicating effective mitigation

T	$w_q = 0$	$w_q = 0.1$	$w_q = 0.2$
1	62.89%	69.95%	71.48%
2	83.93%	86.27%	86.30%
4	91.77%	93.34%	93.73%
8	94.45%	94.93%	95.25%
16	95.22%	95.69%	95.63%
32	95.56%	95.81%	95.78%
64	95.74%	95.87%	95.88%
128	95.79%	95.84%	95.85%

Table 4.1: Test accuracy (%) for VGG-16 on CIFAR-10 across varying time-steps T and QCFS regularization weights w_q.

of unevenness and quantization errors. Beyond $T = 32$, the performance gain saturates, suggesting diminishing returns for high w_q under ample inference time.

4.6 QCFS Regularizer Impact on Low Time-steps

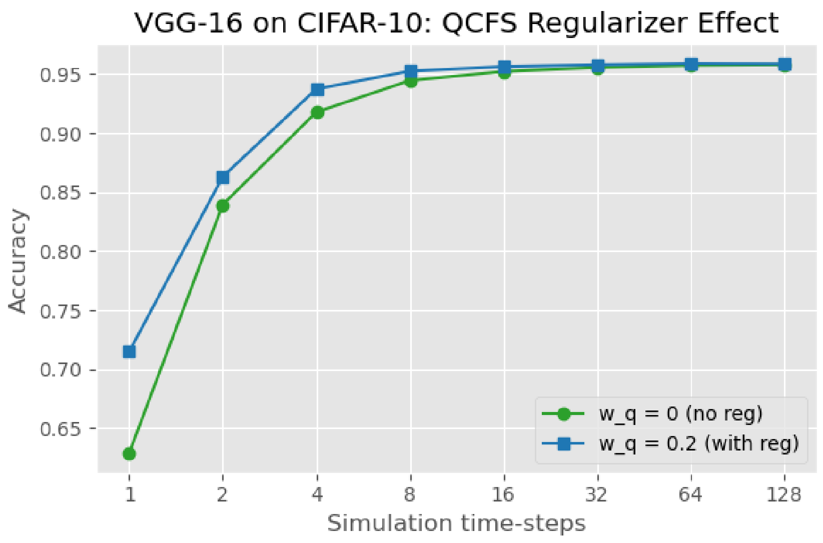


Figure 4.1: Effect of QCFS regularizer on SNN accuracy across simulation time-steps for VGG-16 on CIFAR-10.

At low simulation time-steps, the application of the QCFS regularizer leads to substantial performance gains. As illustrated in Figure 4.1, a regularization weight of $w_q = 0.2$ results in an accuracy increase of over 8% at $T = 1$ compared to the baseline without regularization. This improvement highlights the effectiveness of QCFS regularization in stabilizing spike outputs under low-latency constraints.

4.7 Effect of QCFS Regularizer on Tiny ImageNet

To test whether the QCFS regularizer generalizes beyond CIFAR-10, applied it to a more complex dataset—Tiny ImageNet—using the VGG-16 architecture. The regularization weight was set to $w_q = 0.05$, and the quantization level was fixed at $L = 8$. Accuracy was measured across simulation time-steps $T \in \{1, 2, 4, 8, 16, 32\}$.

T	$w_q = 0$ **(no reg)**	$w_q = 0.05$ **(with reg)**
1	10.71%	11.43%
2	16.67%	17.41%
4	27.67%	28.70%
8	42.62%	43.39%
16	54.41%	55.52%
32	60.10%	60.49%

Table 4.2: VGG-16 accuracy on Tiny ImageNet across time-steps with and without QCFS regularizer.

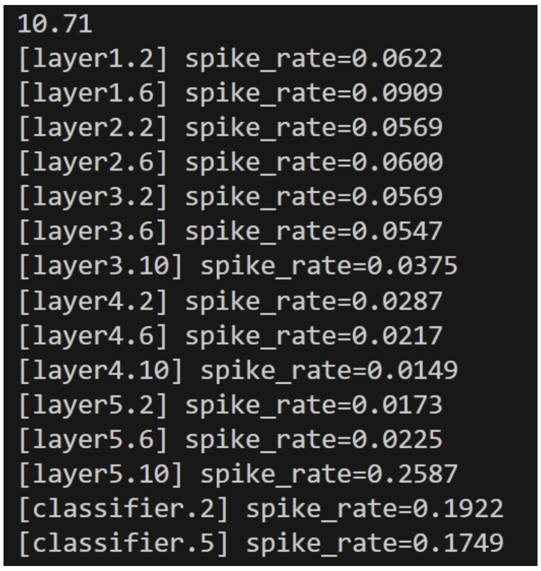
As seen in Table 4.2, applying the QCFS regularizer consistently improves accuracy across all time-steps. However, the improvement is relatively small, especially at higher time-steps and on this more complex dataset. For example, at $T = 1$, accuracy improves by 0.72%, but this benefit gradually reduces with increasing T.

These results suggest that while the QCFS regularizer helps stabilize SNN behavior at low time-steps, it does not significantly boost performance on large or complex datasets.

4.7.1 Effect of QCFS Regularizer on Spike Sparsity

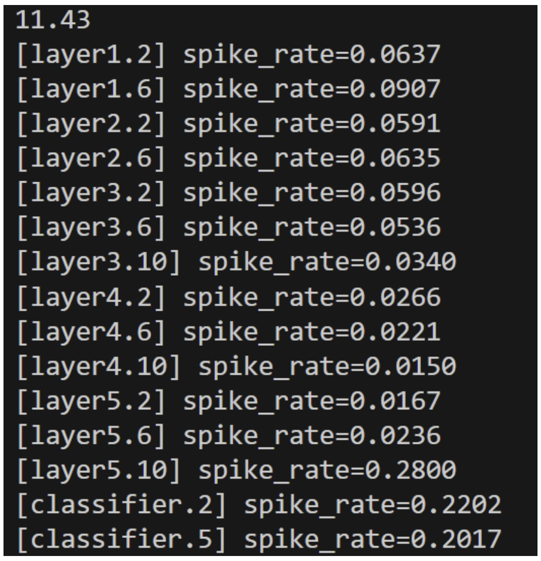
To assess the influence of the QCFS regularizer on spike sparsity, the per-layer spike rates for VGG-16 on Tiny ImageNet were analyzed at time-step $T = 1$.

The figure below compares spike rates in the absence and presence of the QCFS regularizer with $w_q = 0.05$.



```
10.71
[layer1.2] spike_rate=0.0622
[layer1.6] spike_rate=0.0909
[layer2.2] spike_rate=0.0569
[layer2.6] spike_rate=0.0600
[layer3.2] spike_rate=0.0569
[layer3.6] spike_rate=0.0547
[layer3.10] spike_rate=0.0375
[layer4.2] spike_rate=0.0287
[layer4.6] spike_rate=0.0217
[layer4.10] spike_rate=0.0149
[layer5.2] spike_rate=0.0173
[layer5.6] spike_rate=0.0225
[layer5.10] spike_rate=0.2587
[classifier.2] spike_rate=0.1922
[classifier.5] spike_rate=0.1749
```

(a) Per-layer spike rates without QCFS regularizer.



```
11.43
[layer1.2] spike_rate=0.0637
[layer1.6] spike_rate=0.0907
[layer2.2] spike_rate=0.0591
[layer2.6] spike_rate=0.0635
[layer3.2] spike_rate=0.0596
[layer3.6] spike_rate=0.0536
[layer3.10] spike_rate=0.0340
[layer4.2] spike_rate=0.0266
[layer4.6] spike_rate=0.0221
[layer4.10] spike_rate=0.0150
[layer5.2] spike_rate=0.0167
[layer5.6] spike_rate=0.0236
[layer5.10] spike_rate=0.2800
[classifier.2] spike_rate=0.2202
[classifier.5] spike_rate=0.2017
```

(b) Per-layer spike rates with QCFS regularizer.

Figure 4.2: Comparison of per-layer spike rates at $T = 1$, with and without QCFS regularizer.

The QCFS regularizer introduces a moderate increase in spike rates across various layers, particularly in the classifier layers. For instance, the spike rate in `classifier.2` increased from 0.1922 to 0.2202, and in `classifier.5` from 0.1749 to 0.2017. This demonstrates that the QCFS regularizer encourages a more uniform distribution of spikes, thereby reducing extreme sparsity and enhancing representational fidelity in later layers. Additionally, this regularization contributes to improved performance at low time-steps, as shown in earlier sections.

Conclusion

During my internship, I focused on reproducing and extending experiments based on the Quantization Clip–Floor–Shift (QCFS) activation function for converting Artificial Neural Networks (ANNs) to Spiking Neural Networks (SNNs). While the QCFS method had already been proposed in prior research, my work involved reproducing the results through simulations.

An important objective of this project was to examine the effect of a regularizer derived from QCFS. The experiments demonstrated that the regularizer significantly improved performance On simpler datasets such as CIFAR-10 at low time-steps, where spike approximation errors are typically high.

To test the generalization of this method, I extended the experiments to the more complex Tiny ImageNet dataset. The regularizer provided moderate gains in performance, indicating that while it remains effective, its impact is reduced with complex datasets. Additionally, the regularizer was found to reduce extreme sparsity in spikes and promote more balanced firing patterns across network layers.

This project provided me with valuable hands-on experience in neuromorphic computing and deepened my understanding of ANN-to-SNN conversion techniques. It also offered insights into how regularization strategies can improve generalization and stability in low-latency spiking systems.