ANN-to-SNN Conversion

Naman

July 1, 2025

Introduction

- Spiking Neural Networks (SNNs) mimic brain neurons by firing discrete spikes whenever a threshold is crossed, offering significant advantages like low power consumption and fast inference on neuromorphic hardware.
- Traditional Artificial Neural Networks (ANNs) achieve high accuracy and are simpler to train, but they consume more energy.
- Converting ANNs into SNNs combines the advantages of easy ANN training and efficient SNN deployment.
- A major challenge in ANN-to-SNN conversion is achieving high accuracy with fewer computational steps (time-steps); for example, using only $T = 4$ steps drastically reduces performance, limiting real-world usage.

Neuron Models in ANN and SNN

- **ANN Neuron Model:**

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

  - a^l: activations of layer l
  - W^l: weight matrix
  - $h(\cdot)$: ReLU activation

- **SNN Neuron Model (Integrate-and-Fire):**

$$m^l(t) = v^l(t-1) + W^l\, x^{l-1}(t),$$
$$s^l(t) = H\big(m^l(t) - \theta^l\big), \quad v^l(t) = m^l(t) - \theta^l\, s^l(t)$$

  - $v^l(t)$: membrane potential
  - $s^l(t)$: binary spike output
  - θ^l: firing threshold
  - $H(\cdot)$: Heaviside step function

Conversion Errors

There are three types of conversion errors:

1. **Clipping Error**
2. **Quantization Error**
3. **Unevenness Error**

Clipping & Quantization Errors

1. Clipping Error

$$\phi_l(T) \in [0, \theta_l] \quad \text{but} \quad a_l \in [0, a_{l,\max}],$$

so any ANN activation $a_l > \theta_l$ is *clipped* to θ_l in the SNN.

2. Quantization Error

$$\phi_l(T) = \mathrm{clip}\Big(\tfrac{\theta_l}{T} \left\lfloor \tfrac{a_l\, T}{\lambda_l} \right\rfloor,\, 0,\, \theta_l\Big),$$

where $\lfloor \cdot \rfloor$ forces continuous ANN outputs onto discrete spike counts, introducing rounding errors.

Unevenness Error

3. Unevenness Error

Even when the total number of spikes $\sum_t s^l(t)$ matches the ANN activation, their *timing* may be clustered:

$$s^l(t) \text{ concentrated early or late} \quad \Longrightarrow \quad \phi_l(T) \neq W^l\, \phi_{l-1}(T)$$

Non-uniform spike timing alters how downstream neurons integrate inputs, causing discrepancies from the expected rate-based value.

Quantization Clip–Floor–Shift (QCFS) Activation

Standard Quantized Activation

$$\bar{h}(z_l) = \lambda_l \operatorname{clip}\left(\frac{1}{L}\left\lfloor \frac{z_l L}{\lambda_l} \right\rfloor, 0, 1\right).$$

With Shift Term φ

$$\hat{h}(z_l) = \lambda_l \operatorname{clip}\left(\frac{1}{L}\left\lfloor \frac{z_l L}{\lambda_l} + \varphi \right\rfloor, 0, 1\right).$$

- L: quantization steps in ANN
- $\theta_l = \lambda_l$, T: SNN time-steps
- Adding $\varphi = \frac{1}{2}$ *centers* the quantization error distribution

Theorem 1: Zero Error When $T = L$

Theorem

If an ANN uses $\bar{h}$ with $L = T$, $\varphi = 0$, and we convert to an SNN with thresholds $\theta_l = \lambda_l$ and reset-by-subtraction, then

$$\widetilde{\mathrm{Err}}_l = \phi_l(T) - a_l = 0 \quad \textit{for every layer } l.$$

Sketch.

Under these conditions both $\lfloor \cdot \rfloor$ and clipping match exactly between ANN and SNN rates. $\square$

Theorem 2: Zero *Expected* Error for Any T, L

Theorem

Using the shifted QCFS activation with $\varphi = \frac{1}{2}$, then for any T, L, if we set

$$\theta_l = \lambda_l, \quad v_l(0) = \theta_l \, \varphi,$$

the expected conversion error vanishes:

$$\mathbb{E}_z \left[\widetilde{\mathrm{Err}_l} \right] = 0.$$

Sketch.

By modeling z_l uniformly across each quantization bin, the half-step shift symmetrizes rounding so positive and negative errors cancel in expectation. $\square$

Results: VGG-16 on CIFAR-10

- **Quantization steps** $L \in \{4, 8, 16, 32\}$
- **Time-steps** $T \in \{4, 8, 16, 32, 64, 128\}$

$L \backslash T$	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: Accuracy (%) of converted SNN for varying L, T.

Results: ResNet-20 on CIFAR-10

- **Quantization steps** $L \in \{4, 8, 16, 32\}$
- **Time-steps** $T \in \{4, 8, 16, 32, 64, 128\}$

$L \backslash T$	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: Accuracy (%) of converted SNN for varying L, T.

QCFS Activation and Regularizer

QCFS Activation

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

QCFS Regularization Loss

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

Mechanism

1. **Clip** activations to $[0, \lambda]$.
2. **Floor** with shift $+0.5$ to achieve nearest-bin rounding.
3. **Shift back** and scale, ensuring zero-mean rounding error.
4. Regularizer penalizes $(z - q)^2$, pulling z to bin centers.

Experimental Setup

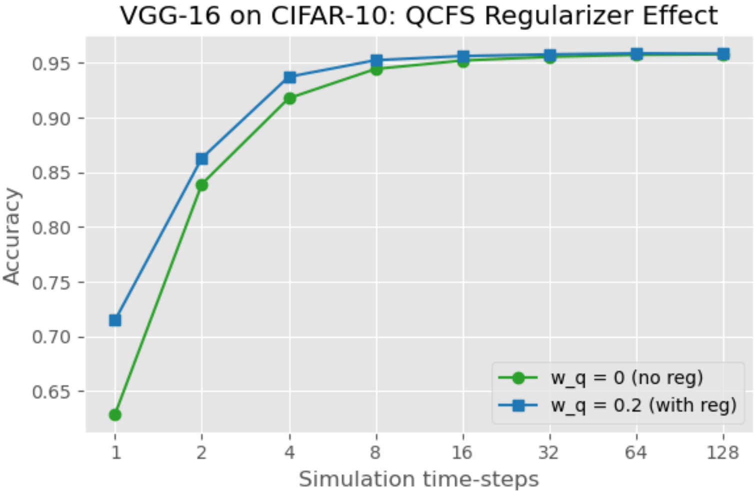
- **Model:** VGG16 on CIFAR-10
- **QCFS weights:** $w_q = 0.1, 0.2, 0.3$
- **Quantization levels:** $L = 8$
- **Time-steps:** $T = 1, 2, 4, 8, 16, 32, 64, 128$

Accuracy vs. Time-steps

T	$\boldsymbol{w_q = 0}$	$\boldsymbol{w_q = 0.1}$	$\boldsymbol{w_q = 0.2}$	$\boldsymbol{w_q = 0.3}$
1	62.89%	69.95%	71.48%	71.05%
2	83.93%	86.27%	86.30%	87.32%
4	91.77%	93.34%	93.73%	93.30%
8	94.45%	94.93%	95.25%	95.17%
16	95.22%	95.69%	95.63%	95.66%
32	95.56%	95.81%	95.78%	95.86%
64	95.74%	95.87%	95.88%	–
128	95.79%	95.84%	95.85%	–

Key Observations

- **Low-T boost:** The QCFS regularizer yields over 8% absolute accuracy gain at $T = 1$ compared to baseline.



Tiny ImageNet: VGG-16 QCFS Regularizer Effect

T	$\boldsymbol{w_q = 0}$ (no reg)	$\boldsymbol{w_q = 0.05}$ (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: VGG-16 accuracy on Tiny ImageNet across time-steps.

Per-Layer Spike Rates (T=1)

Without QCFS Regularizer

```
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
```

With 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
```

- slightly shifts some layer rates (e.g. classifier layers increase from 19.2%→20.2%).

Thank You

Thank You!