Supplementary Materials: Towards High-performance Spiking Transformers from ANN to SNN Conversion

Anonymous Authors

1 PROOF OF THEOREM 1

Theorem 1.1. *Consider a non-linear layer l with a function F. In SNNs, the output of this layer at time t is denoted as $O^l(t)$. Let $S^l(T)$ be the cumulative sum of layer l outputs up to time T, given by $S^l(T) = \sum_{t=1}^{T} O^l(t)$. The expected output of the SNNs at time T is given by:*

$$O^l(T) = TF\left(\frac{S^{l-1}(T)}{T}\right) - (T-1)F\left(\frac{S^{l-1}(T-1)}{T-1}\right). \quad (1)$$

Proof. According to Section 3.2, we denote $x^l(t)$ as $O^l(t)$, which has the same meaning, and we can approximate the output value of ANNs using the mean value of the output for the first T time steps in SNNs:

$$a_T^l = \Phi^l(T) = \frac{\sum_{t=1}^{T} O^l(t)}{T} \quad (2)$$

where a_T^l represents the estimated values of neurons in layer l at time T in ANNs. It will change as the corresponding spikes in SNNs accumulate over time.

Meanwhile, in the case of ANNs, a_T^l can be formulated as:

$$a_T^l = F(a_T^{l-1}). \quad (3)$$

Furthermore, we can deduce the output by subtracting the total output of the previous T and T-1 time steps from the formula 2 and 3.

$$
\begin{aligned}
O^l(T) &= \sum_{t=1}^{T} O^l(t) - \sum_{t=1}^{T-1} O^l(t) \\
&= Ta_T^l - (T-1)a_{T-1}^l \\
&= TF(a_T^{l-1}) - (T-1)F(a_{T-1}^{l-1}) \\
&= TF\left(\frac{\sum_{t=1}^{T} O^{l-1}(t)}{T}\right) - (T-1)F\left(\frac{\sum_{t=1}^{T-1} O^{l-1}(t)}{T-1}\right) \\
&= TF\left(\frac{S^{l-1}(T)}{T}\right) - (T-1)F\left(\frac{S^{l-1}(T-1)}{T-1}\right)
\end{aligned}
\quad (4)
$$

$\square$

2 PROOF OF THEOREM 2

Theorem 2.1. *Consider a module for matrix product that receives two sets of spike inputs, denoted by $A_{v_a}(t)$ and $B_{v_b}(t)$. These inputs are generated by neurons A and B, respectively, and are characterized by multiple thresholds v_a and v_b, as described in Section 4.3.*

We can integrate the input by $A(t) = \sum_{v_a} v_a A_{v_a}(t)$ and $B(t) = \sum_{v_b} v_b B_{v_b}(t)$. Here, $A(t)$ and $B(t)$ are the sum matrices weighted by multiple thresholds v_a and v_b, respectively.

Let $S_A(T) = \sum_{t=1}^{T} A(t)$ and $S_B(T) = \sum_{t=1}^{T} B(t)$ represent the cumulative sum of inputs up to time T. We define $S_K(T) = S_A(T)S_B(T)$.

Then, the expected output at time T can be formulated as:

$$O(T) = \frac{1}{T}S_K(T) - \frac{1}{T-1}S_K(T-1), \quad (5)$$

where $S_K(T)$ can be calculated mainly using addition, as described by the following equation:

$$S_K(T) = S_K(T-1) + K(T) \quad (6)$$

$$
\begin{aligned}
K(T) = &\sum_{v_a,v_b} v_a v_b A_{v_a}(T)B_{v_b}(T) + \sum_{v_a} v_a A_{v_a}(T)S_B(T-1) \\
&+ \sum_{v_b} v_b S_A(T-1)B_{v_b}(T).
\end{aligned}
\quad (7)
$$

Proof. Since we approximate the value of ANNs using the mean value for the first T times in SNNs, let the expected input matrices A_T, B_T, and $O_T = A_T B_T$ in ANNs be calculated based on the input spikes during the first T time steps in SNNs, denoted as:

$$A_T = \frac{\sum_{t=1}^{T} A(t)}{T} \quad (8)$$

$$B_T = \frac{\sum_{t=1}^{T} B(t)}{T} \quad (9)$$

$$O_T = \frac{\sum_{t=1}^{T} O(t)}{T} \quad (10)$$

So, the expected output matrix $O(T)$ at time T can be calculated by:

$$
\begin{aligned}
O(T) &= \sum_{t=1}^{T} O(t) - \sum_{i=t}^{T-1} O(t) \\
&= TO_T - (T-1)O_{T-1} \\
&= TA_T B_T - (T-1)A_{T-1}B_{T-1} \\
&= T\frac{\sum_{t=1}^{T} A(t)}{T}\frac{\sum_{t=1}^{T} B(t)}{T} \\
&\quad - (T-1)\frac{\sum_{t=1}^{T-1} A(t)}{T-1}\frac{\sum_{t=1}^{T-1} B(t)}{T-1} \\
&= \frac{1}{T}\sum_{t=1}^{T} A(t)\sum_{t=1}^{T} B(t) - \frac{1}{(T-1)}\sum_{t=1}^{T-1} A(t)\sum_{t=1}^{T-1} B(t) \\
&= \frac{1}{T}S_A(T)S_B(T) - \frac{1}{T-1}S_A(T-1)S_B(T-1) \\
&= \frac{1}{T}S_K(T) - \frac{1}{T-1}S_K(T-1)
\end{aligned}
\quad (11)
$$

And $S_K(T)$ can be calculated by:

$$S_K(T) = S_A(T)S_B(T)$$

$$= (\sum_{t=1}^{T} A(t))(\sum_{t=1}^{T} B(t))$$

$$= (S_A(T-1) + A(T))(S_B(T-1) + B(T))$$

$$= S_A(T-1)S_B(T-1) + A(T)B(T)$$

$$+ A(T)S_B(T-1) + S_A(T-1)B(T)$$

$$= S_K(T-1) + \sum_{v_a,v_b} v_a v_b A_{v_a}(T)B_{v_b}(T)$$

$$+ \sum_{v_a} v_a A_{v_a}(T)S_B(T-1) + \sum_{v_b} v_b S_A(T-1)B_{v_b}(T)$$

$$= S_K(T-1) + K(T). \qquad (12)$$

Assuming the dimension of $S_K(T), S_A(T)$ and $S_B(T)$ are $n \times m$, $n \times p$ and $p \times m$, respectively. And suppose the firing rate of $A(T)$ and $B(T)$ are η_1 and η_2.

In order to determine the number of different operations required to update $S_K(T)$, we conduct a brief analysis: Multiplications occur when the threshold is multiplied by the results of various matrix multiplications; Additions occur during the calculation of individual matrix multiplications, as well as the accumulation of the results of the four parts.

As each position of the input matrix has only one effective threshold at each time, it restricts the total number of input spikes, thus limiting the total number of operations.

The maximum addition operation number is

$$ACs_{\text{SNN}}^{max} = \eta_1 \eta_2 npm + \eta_1 npm + \eta_2 npm + 3nm \qquad (13)$$

where $\eta_1 \eta_2 npm$, $\eta_1 npm$ and $\eta_2 npm$ are the maximum addition operations in calculating $\sum_{v_a,v_b} v_a v_b A_{v_a}(T)B_{v_b}(T)$, $\sum_{v_a} v_a A_{v_a}(T)S_B(T-1)$ and $\sum_{v_b} v_b S_A(T-1)B_{v_b}(T)$, respectively. $3nm$ is the maximum operation in accumulating four parts in Equation (6).

The maximum multiplication operation number is

$$ACs_{\text{SNN}}^{max} = min(\eta_1, \eta_2)nm + \eta_1 nm + \eta_2 nm \qquad (14)$$

where $min(\eta_1, \eta_2)nm$, $\eta_1 nm$ and $\eta_2 nm$ are the maximum multiplication operations in calculating $\sum_{v_a,v_b} v_a v_b A_{v_a}(T)B_{v_b}(T)$, $\sum_{v_a} v_a A_{v_a}(T)S_B(T-1)$ and $\sum_{v_b} v_b S_A(T-1)B_{v_b}(T)$, respectively.

It can be seen that $ACs_{\text{SNN}}^{max} \gg MACs_{\text{SNN}}^{max}$, so $S_K(T)$ can be calculated mainly using addition. □

3 EXPERIMENT DETAILS

3.1 Datasets

CIFAR-10. The CIFAR-10 dataset [7] consists of 60000 32×32 images in 10 classes. There are 50000 training images and 10000 test images.

CIFAR-100. The CIFAR-100 dataset [7] consists of 60000 32×32 images in 100 classes. There are 50000 training images and 10000 test images.

ImageNet1k. We use the ILSVRC 2012 dataset [9], which consists of 1,281,167 training images and 50000 testing images.

3.2 Data Preprocessing

To process our image data, we followed a series of steps. First, we resized the image to the desired size and then cropped it to match the input size. After that, we converted the image into a PyTorch tensor. Next, we normalized the pixel values using the provided mean and standard deviation values. The mean and standard deviation values were specified as (0.48145466, 0.4578275, 0.40821073) and (0.26862954, 0.26130258, 0.27577711). Finally, we normalized the pixel values of the three-channel images based on the provided mean and standard deviation.

3.3 Experimental Setup

The conversion in this paper is based on pre-trained Vision Transformer including the ViT-S/16, ViT-B/16, ViT-L/16 with 224 resolution [10], and the EVA model eva_g_patch14 in [3].

For all Multi-Threshold Neurons, we set n to 8 for ViT-S/16, ViT-B/16, ViT-L/16 and 6 for EVA. We set threshold percent p to 99 to get thresholds for each neuron. In particular, due to huge differences in GELU and softmax layers' output values, we configure the positive and negative base thresholds to 0.5 and 0.08, respectively, for neurons following the GELU module in ViT models, and to 0.0125 for neurons following the softmax module to prevent too few spikes.

Besides, the precision of the network is highly sensitive to the precision of the classification layer, as mentioned in [8]. Since the classification layer has minimal energy consumption during runtime, we retained analog input in the classification layer.

4 ADDITIONAL EXPERIMENTAL DETAILS

4.1 Detailed results on other datasets

Tables 1 and 2 present a comparison of the accuracy and energy consumption of different neural network architectures - ANNs and SNNs - on CIFAR10 and CIFAR100 datasets.

Table 1 compares the accuracy of ANN and SNN architectures for the CIFAR10 dataset across three model scales: ViT-S/16, ViT-B/16, and ViT-L/16. It can be seen that the SNN model can reach a comparable accuracy while significantly reducing the consumption. For example, when the SNN model is run for 6 time steps, models such as ViT-S/16, ViT-B/16, and ViT-L/16 achieve accuracy levels of 97.37%, 98.24%, and 99.1%, respectively. The remarkable fact is that they only consume 0.6, 0.48, and 0.4 energy, respectively when compared to the original ANN (Artificial Neural Network) models.

Table 2 presents a similar comparison for the more complex CIFAR100 dataset. For instance, at 6 timesteps, ViT-S/16, ViT-B/16, and ViT-L/16 achieve accuracies of 84.75%, 90.22%, and 93.04%, respectively, while using only 0.61, 0.48, and 0.43 energy compared to original ANN models. It shows the potential of our method to reduce energy consumption while maintaining accuracy. The results demonstrate our method's potential to reduce energy consumption while maintaining accuracy.

Table 1: Accuracy and energy consumption ratio of ECMT(Ours) on CIFAR10 dataset

Arch.	Accuracy/Energy	Original (ANN)	Ours (SNN)					
			T=1	T=2	T=4	T=6	T=8	T=10
ViT-S/16	Acc. (%)	98.33	8.53	31.32	93.82	97.37	98.01	98.21
	Energy ratio	1	0.06	0.15	0.37	0.60	0.82	1.03
ViT-B/16	Acc. (%)	98.75	9.17	32.25	95.17	98.24	98.55	98.69
	Energy ratio	1	0.04	0.12	0.30	0.48	0.66	0.83
ViT-L/16	Acc. (%)	99.07	10.55	95.14	98.89	99.1	99.03	99.08
	Energy ratio	1	0.03	0.11	0.27	0.42	0.57	0.72

Table 2: Accuracy and energy consumption ratio of ECMT(Ours) on CIFAR100 dataset

Arch.	Accuracy/Energy	Original (ANN)	Ours (SNN)					
			T=1	T=2	T=4	T=6	T=8	T=10
ViT-S/16	Acc. (%)	89.28	0.95	4.9	69.49	84.75	87.83	88.93
	Energy ratio	1	0.06	0.16	0.38	0.61	0.84	1.07
ViT-B/16	Acc. (%)	92.26	0.87	17.07	82.86	90.22	91.5	91.91
	Energy ratio	1	0.04	0.12	0.30	0.48	0.66	0.84
ViT-L/16	Acc. (%)	93.84	1.61	69.08	91.82	93.04	93.34	93.56
	Energy ratio	1	0.04	0.12	0.27	0.43	0.58	0.73

4.2 Comparison with the State-of-the-art on CIFAR10 and CIFAR100 datasets

We compare the experimental results using the ViT-S/16, ViT-B/16, ViT-L/16 model on the CIFAR10 and CIFAR100 datasets with previous state-of-the-art methods, as shown in Table 3 and 4.

In the evaluation of the CIFAR10 dataset, the ECMT model achieved an impressive accuracy rate of 97.37%, 98.24%, and 99.1% respectively, using the architecture of ViT-S/16, ViT-B/16, ViT-L/16 over just six timesteps. This level of precision is highly competitive, especially compared to similarly-sized models. In evaluating the CIFAR100 dataset, considered more complex, the ECMT method again displays its strength. The results demonstrate that the ECMT method achieves a similar high accuracy.

The ECMT model uses the Transformer-to-SNN approach and has performed exceptionally well on the CIFAR10 and CIFAR100 datasets. Its ViT-B/16 variant stands out by achieving high accuracy with a moderate number of parameters, indicating the potential of SNNs in achieving state-of-the-art results with a significant reduction in computational resources. This balance of efficiency and accuracy makes the ECMT a promising model for energy-efficient and fast processing tasks.

Table 3: Comparison between the proposed method and previous works on CIFAR10 dataset

Method	Type	Arch.	Param. (M)	T	Accuracy (%)
Spikingformer[14]	Direct Training	Spikingformer-4-384-400E	9.32	4	95.81
Spike-driven Transformer[13]	Direct Training	Spikingformer-4-384-400E	9.32	4	95.6
RMP[4]	CNN-to-SNN	VGG-16	138	64(2048)	90.35(93.63)
SNM[11]	CNN-to-SNN	VGG-16	138	32(128)	93.43(94.07)
TS[2]	CNN-to-SNN	VGG-16	138	16(32)	92.29(92.29)
QFFS[8]	CNN-to-SNN	VGG-16	138	4	92.64
QCFS[1]	CNN-to-SNN	ResNet-18	11.8	8(64)	94.82(96.06)
		VGG-16	138	8(64)	94.95(95.55)
SRP[5]	CNN-to-SNN	ResNet-18	11.8	4(16)	95.25(95.55)
		VGG-16	138	4(16)	95.32(95.42)
MST[12]	Transformer-to-SNN	Swin-T(BN)	27.6	64(256)	96.32(97.27)
STA[6]	Transformer-to-SNN	ViT-B/32	86	32(256)	95.49(95.82)
ECMT(Ours)	Transformer-to-SNN	ViT-S/16	22	6(8)	97.37(98.01)
		ViT-B/16	86	6(8)	98.24(98.55)
		ViT-L/16	307	6(8)	99.1(99.03)

Table 4: Comparison between the proposed method and previous works on CIFAR100 dataset

Method	Type	Arch.	Param. (M)	T	Accuracy (%)
Spikingformer[14]	Direct Training	Spikingformer-4-384-400E	9.32	4	79.21
Spike-driven Transformer[13]	Direct Training	Spikingformer-4-384-400E	9.32	4	78.4
RMP[4]	CNN-to-SNN	VGG-16	138	128(2048)	63.76(70.93)
SNM[11]	CNN-to-SNN	VGG-16	138	32(128)	71.8(73.95)
TS[2]	CNN-to-SNN	VGG-16	138	16(64)	63.73(69.27)
QCFS[1]	CNN-to-SNN	ResNet-18	11.8	8(64)	78.48(79.54)
		VGG-16	138	8(64)	73.96(77.10)
SRP[5]	CNN-to-SNN	ResNet-20	0.27	4(32)	59.34(65.50)
		VGG-16	138	4(32)	75.42(76.45)
MST[12]	Transformer-to-SNN	Swin-T(BN)	27.6	64(256)	85.4(86.91)
STA[6]	Transformer-to-SNN	ViT-B/32	86	32(256)	84.15(85.98)
ECMT(Ours)	Transformer-to-SNN	ViT-S/16	22	6(8)	84.75(87.83)
		ViT-B/16	86	6(8)	90.22(91.5)
		ViT-L/16	307	6(8)	93.04(93.34)

REFERENCES

[1] Tong Bu, Wei Fang, Jianhao Ding, PENGLIN DAI, Zhaofei Yu, and Tiejun Huang. 2022. Optimal ANN-SNN Conversion for High-accuracy and Ultra-low-latency Spiking Neural Networks. In *International Conference on Learning Representations*.

[2] Shikuang Deng and Shi Gu. 2021. Optimal Conversion of Conventional Artificial Neural Networks to Spiking Neural Networks. In *International Conference on Learning Representations*.

[3] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. 2023. EVA: Exploring the Limits of Masked Visual Representation Learning at Scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19358–19369.

[4] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. 2020. RMP-SNN: Residual Membrane Potential Neuron for Enabling Deeper High-Accuracy and Low-Latency Spiking Neural Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13558–13567.

[5] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. 2023. Reducing ANN-SNN Conversion Error through Residual Membrane Potential. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 1 (2023), 11–21.

[6] Yizhou Jiang, Kunlin Hu, Tianren Zhang, Haichuan Gao, Yuqian Liu, Ying Fang, and Feng Chen. 2024. Spatio-Temporal Approximation: A Training-Free SNN Conversion for Transformers. In *Proceedings of the International Conference on Learning Representations*.

[7] A Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront* (2009).

[8] Chen Li, Lei Ma, and Steve Furber. 2022. Quantization Framework for Fast Spiking Neural Networks. *Frontiers in Neuroscience* 16 (2022), 918793.

[9] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision* 115 (2015), 211–252.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30.

[11] Yuchen Wang, Malu Zhang, Yi Chen, and Hong Qu. 2022. Signed Neuron with Memory: Towards Simple, Accurate and High-Efficient ANN-SNN Conversion. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 2501–2508.

[12] Ziqing Wang, Yuetong Fang, Jiahang Cao, Qiang Zhang, Zhongrui Wang, and Renjing Xu. 2023. Masked Spiking Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1761–1771.

[13] Man Yao, JiaKui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. 2023. Spike-driven Transformer. In *Advances in Neural Information Processing Systems*, Vol. 36. 64043–64058.

[14] Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Han Zhang, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. 2023. Spikingformer: Spike-driven Residual Learning for Transformer-based Spiking Neural Network. *arXiv preprint arXiv:2304.11954* (2023).