SEENN:
Towards temporal spiking early-exit neural networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Spiking Neural Networks (SNNs) have recently become more popular as a biologically plausible substitute for traditional Artificial Neural Networks (ANNs). SNNs are cost-efficient and deployment-friendly because they process input in both spatial and temporal manner using binary spikes. However, we observe that the information capacity in SNNs is affected by the number of timesteps, leading to an accuracy-efficiency tradeoff. In this work, we study a fine-grained adjustment of the number of timesteps in SNNs. Specifically, we treat the number of timesteps as a variable conditioned on different input samples to reduce redundant timesteps for certain data. We call our method **S**piking **E**arly-**E**xit **N**eural **N**etworks (**SEENNs**). To determine the appropriate number of timesteps, we propose SEENN-I which uses a confidence score thresholding to filter out the uncertain predictions, and SEENN-II which determines the number of timesteps by reinforcement learning. Moreover, we demonstrate that SEENN is compatible with both the directly trained SNN and the ANN-SNN conversion. By dynamically adjusting the number of timesteps, our SEENN achieves a remarkable reduction in the average number of timesteps during inference. For example, our SEENN-II ResNet-19 can achieve **96.1**% accuracy with an average of **1.08** timesteps on the CIFAR-10 test dataset.

1 Introduction

Deep learning has revolutionized a range of computational tasks such as computer vision and natural language processing [1] using Artificial Neural Networks (ANNs). These successes, however, have come at the cost of tremendous computational demands and high latency [2]. In recent years, Spiking Neural Networks (SNNs) have gained traction as an energy-efficient alternative to ANNs [3,4]. SNNs infer inputs across a number of timesteps as opposed to ANNs, which infer over what is essentially a single timestep. Moreover, during each timestep, the neuron in an SNN either fires a spike or remains silent, thus making the output of the SNN neuron binary and sparse. Such spike-based computing produces calculations that substitute multiplications with additions.

In the field of SNN research, there are two main approaches to getting an SNN: (1) directly training SNNs from scratch and (2) converting ANNs to SNNs. Direct training seeks to optimize an SNN using methods such as spike timing-based plasticity [5] or surrogate gradient-based optimization [6,7]. In contrast, the ANN-SNN conversion approach [8, 9, 10, 11, 12, 13] uses the feature representation of a pre-trained ANN and aims to replicate it in the corresponding SNN. Both methods have the potential to achieve high-performance SNNs when implemented correctly.

Despite the different approaches, both training-based and conversion-based SNNs are limited by binary activations. As a result, the key factor that affects their information processing capacity is the **number of timesteps**. Expanding the number of timesteps enables SNNs to capture more features in the temporal dimension, which can improve their accuracy in conversion and training. However, a

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

A Verification of Assumption 3.1

In Assumption 3.1, we conjecture that if a correct prediction is made by $f_t(\mathbf{x})$, then for all $t' \geq t$ that $f_{t'}(\mathbf{x})$ is also correct. And based on this assumption, we propose the equation to compute AET, *i.e.* Eq. (4). Here, to validate if our assumption holds, we propose another metric: *empirical AET*, given by

$$\widetilde{\text{AET}} = \frac{1}{N}\left(\sum_{i=1}^{N} \tilde{t}_i\right), \tag{13}$$

where $\tilde{t}_i$ is the actual earliest number of timesteps that predicts the correct class for the i-th sample. Note that, similar to AET, if the network cannot make correct predictions in any timesteps, then we set $\tilde{t}_i$ to the maximum number of timesteps.

Consider an example that does not satisfy Assumption 3.1, for example, the prediction results for the first 4 timesteps are {False, True, False, True}. The empirical AET will obtain 2 while the AET will obtain $2 - 3 + 4 = 3$. Therefore, by comparing the difference between the AET and the empirical AET we can verify if Assumption 3.1 holds. Table 5, as shown below, demonstrates the AET and the empirical AET comparison across different models and datasets. We can find that difference is very small, often less than 0.05. The only dataset that creates a slightly larger difference is CIFAR10-DVS. Indeed, the event-stream dataset may provide more variations over time. Nevertheless, the potential for early exit is still high.

Table 5: Comparison between ATE and empirical AET across models and datasets.

Model	Dataset	$\max T$	AET	$\widetilde{\text{AET}}$
	Direct Training of SNNs			
ResNet-19	CIFAR-10	4	1.1309	1.1188
ResNet-19	CIFAR-100	4	1.6075	1.5616
ResNet-34	ImageNet	6	2.9026	2.7781
SEW-ResNet-34	ImageNet	4	2.0646	2.0092
VGGSNN	CIFAR10-DVS	10	3.096	2.774

B Hardware Evaluation

For latency measurement, we directly tested on GPU with the Pytorch framework. The latency measurement is computed across the whole test dataset. For example, the throughput calculation is given by

$$\text{Throughput} = \frac{\text{Test set inference time}}{\text{Number of test samples}}. \tag{14}$$

For energy estimation, we adopt the conventional way to measure the addition and multiplication operations for the entire SNN inference. The addition costs $0.9pJ$ and the multiplication costs $4.6pJ$, respectively.

C Architecture Details

The major network architecture we adopt in this work is ResNet-series architecture. There are many variants of ResNets used in the field of SNNs, *e.g.* ResNet-18, ResNet-19, and ResNet-20. They are also mixedly referred to in existing literature [17, 19, 26, 49]. Therefore, to avoid confusion in these ResNet architectures, in this section, we sort out the details of the network configurations.

Table 6 summarizes the different ResNets we used in our CIFAR experiments. For ImageNet models, we use the standard ResNet-34 [49] and SEW-ResNet-34 [52] which are well-defined. The basic difference between the four ResNets is the channel configurations and the block configurations. Therefore, their actual FLOPs difference is a lot larger than the difference suggested by their names. For example, our policy network (ResNet-8) only contains 0.547% number of operations of ResNet-19. Therefore, the cost of running SEENN-II is almost negligible.

Table 6: The architecture details of ResNets. * denotes that the first residual block contains downsample layer to reduce the feature resolution.

	ResNet-8	**ResNet-18**
conv1	$3 \times 3, 16, s1$	$3 \times 3, 64, s1$
stage1	$\begin{pmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{pmatrix} \times 1$	$\begin{pmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{pmatrix} \times 2$
stage2	$\begin{pmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{pmatrix}^* \times 1$	$\begin{pmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{pmatrix}^* \times 2$
stage3	$\begin{pmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{pmatrix}^* \times 1$	$\begin{pmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{pmatrix}^* \times 2$
stage4	N/A	$\begin{pmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{pmatrix}^* \times 2$
pooling	Global average pooling	
classifier	$(64, O)$ FC	$(512, O)$ FC
	ResNet-19	**ResNet-20**
conv1	$3 \times 3, 128, s1$	$3 \times 3, 16, s1$
stage1	$\begin{pmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{pmatrix} \times 3$	$\begin{pmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{pmatrix} \times 2$
stage2	$\begin{pmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{pmatrix}^* \times 3$	$\begin{pmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{pmatrix}^* \times 2$
stage3	$\begin{pmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{pmatrix}^* \times 2$	$\begin{pmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{pmatrix}^* \times 2$
pooling	Global average pooling	
classifier	$(512, O)$ FC	$(64, O)$ FC