IM-SNN: Memory-Efficient Spiking Neural Network with Low-Precision Membrane Potentials and Weights

Ahmed Hasssan
School of ECE
Cornell Tech
New York, NY, USA
ah2288@cornell.edu

Jian Meng
School of ECE
Cornell Tech
New York, NY, USA
jm2787@cornell.edu

Anupreetham Anupreetham
School of ECE
Cornell Tech
New York, NY, USA
aa2483@cornell.edu

Jae-sun Seo
School of ECE
Cornell Tech
New York, NY, USA
js3528@cornell.edu

***Abstract*—Compared to the conventional deep neural network (DNN), the binary activation (spikes) of spiking neural network (SNN) largely improves energy efficiency, especially with spatial-temporal computer vision tasks. Despite the memory reduction introduced by the binary spikes, the full precision membrane potential of SNNs requires the same-sized memory array as the full precision activation of DNNs. In other words, the binary spikes of SNNs elevate the efficiency of layer-wise communication but do not necessarily improve the total storage or overall memory efficiency. Prior research works have not fully investigated low-precision membrane potential considering the overall SNN memory efficiency and hardware awareness. Motivated by that, this paper proposes *IM-SNN*, an integer-only SNN with low-precision weights and membrane potential. Unlike prior works that ignore the hardware bottleneck of iterative membrane updates, IM-SNN provides a systematic compression scheme that makes the quantized SNN fully deployable to the hardware accelerator. In addition to the low-precision weights, IM-SNN compresses the membrane potential down to ternary precision, leading to outstanding hardware compatibility and memory efficiency. The proposed method has been validated against both static image datasets (CIFAR, ImageNet) and event-based datasets (DVS-CIFAR10, N-Caltech, Prophesee Automotive Gen1) for both classification and object detection tasks. Compared to the state-of-the-art (SoTA) SNN works, our work achieves up to 13× memory reduction with negligible accuracy degradation.**

***Index Terms*—Spiking neural network (SNN), low-precision quantization, membrane potential, efficient SNN.**

I. Introduction

Spiking neural networks (SNN) have been presented as efficient AI models for processing spatial-temporal information. Empowered by the revolution of deep neural networks (DNNs), recent SNNs started applying the spiking procedure on DNN with spike-based spatial-temporal computer vision tasks (e.g., event-based computer vision). As the key ingredient of SNN, binary spikes largely elevate the efficiency of neural networks with reduced memory consumption and simplified hardware design. Meanwhile, the non-differentiability of binary activation required prior works to investigate the high-performance training methods and mitigate the accuracy gap between SNN and DNN on large-scale datasets (e.g., ImageNet-1K). Besides the accuracy-driven objective, the ultimate goal of SNN is employing the algorithm-induced efficiency to **practical hardware benefits**. Although the binarized spikes improve the efficiency of computation, generating the binarized spikes requires **high-precision** membrane potential with iterative updates. Regardless of the training methods, the layer-wise operation of the current SNN design consists of a) high precision weights, b) high precision membrane potential, and c) input binary spikes. Compared to the full-precision DNN, the binary spikes alleviate the complexity of the SNN operation itself, but the full-precision membrane potential can require a large amount of memory storage, as illustrated in Figure 1. Furthermore, the iterative updates with the high-precision membrane potential require frequent memory accesses, deteriorating the latency and overall energy efficiency. Unfortunately, such bottlenecks have not been fully investigated in prior SNN works [1]–[3].

Motivated by this, representing the membrane potential with low precision becomes a critical step to enhance the hardware efficiency of SNNs, towards executing SNN inference for energy-/area-constrained edge computing. Some of the prior works, Q-SpiSNN [4], and Loihi [5] have used 12-bit and 24-bit fixed-point membrane potential representation with low precision of weights to optimize the SNN hardware resources. However, such works still use relatively high precision for membrane potential compared to the mainstream quantization scheme (e.g., INT8). Furthermore, the accuracy of Q-SpiSNN [4] failed to achieve comparable performance as the recent SoTA SNN methods across a wide spectrum of vision tasks.

To bridge these research gaps, we propose *IM-SNN*, a fully-quantized spiking neural network with *ultra-low precision weights and membrane potential*, enabling the end-to-end low precision operations on SNN with outstanding memory efficiency and negligible accuracy degradation compared to the SoTA SNN baseline. It should be noted that compressing the entire SNN operation to low precision is not equivalent to employing quantizers everywhere. In fact, compressing the membrane potential to low precision is even more challenging

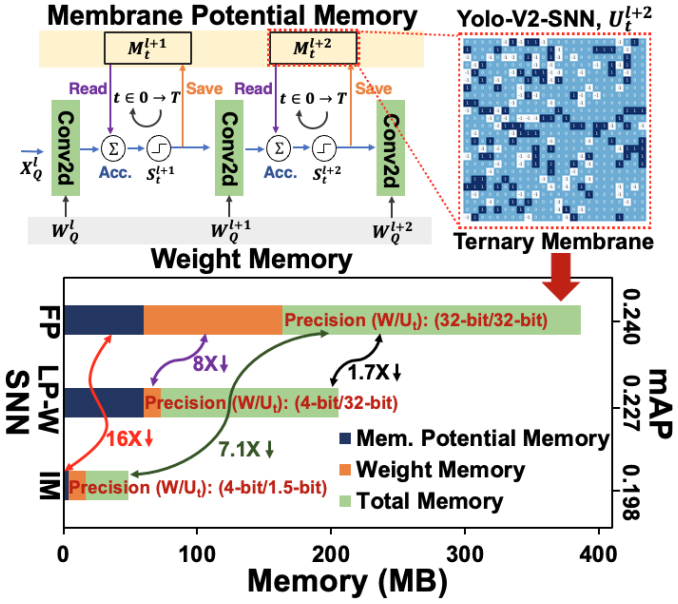


Fig. 1: (a) Proposed IM-SNN algorithm with memory cost reduction of IM-SNN with YOLO-V2 on Prophesse Gen 1.

compared to the traditional DNN-based quantization. We first identify the following challenges for low-precision membrane potential:

a) Optimal quantization boundary: One of the major design points of the quantization algorithm is to allocate the optimal clipping boundary in the full precision distribution (e.g., weight or activation) of DNN. However, the membrane potential of SNN is iteratively updated at different time steps during inference. Therefore, finding the optimal clipping boundary is critical for membrane potential quantization.

b) Incompatibility of iterative dequantization: The standard quantization process includes the "quantize and dequantize" workflow to scale the low precision representation (e.g., INT8) back to the high precision floating point range. The quantized integer and the dequantized floating point data are connected by the high-precision scaling factor. As demonstrated in [6], post-quantization scaling is required to avoid the mismatched numerical range. In SNN, the iteratively updated membrane potential for every neuron requires continuous quantization after each time step to maintain the low precision. However, rescaling the updated membrane potential at **each** time step requires repetitive high-precision multiplication, magnifying the cost of hardware implementation. Therefore, the traditional quantization-aware training (QAT) scheme is incompatible with low-precision membrane potential. Achieving the integer-only computation on SNN requires a dedicated quantization scheme and training method.

The proposed *IM-SNN* work addresses the aforementioned challenges with the following key contributions:

- This paper first investigates the impact of membrane potential clipping towards optimizing the clipping boundary. The comprehensive empirical analysis characterizes the relationship between membrane potential clipping and neuron spiking.
- This paper proposes a novel quantization-aware training algorithm designed for integer-only SNN. In particular, we propose Stacked Gradient Surrogation (SGS), a novel SNN training scheme designed for low-precision membrane potential training.
- The proposed IM-SNN achieves up to 8.13× and 13× total inference memory reduction on complex static image datasets (CIFAR, ImageNet) and event datasets (DVS-CIFAR10, N-Caltech, Prophesee Gen1) with various deep SNN architectures.

II. Related Works

a) Direct SNN Training: Recent SNN works focus on direct SNN training from scratch with various gradient approximations and computation. To overcome the non-differentiable bottleneck of the spiking function, BNTT [7] introduces the gradient surrogation, which approximates the gradient landscape by the designed non-linear function (e.g., Sigmoid). Driven by accuracy and different model architectures, various SG functions have been proposed, such as rectangle function, arctangent, and triangle functions [8]. DSpike [9] introduces a non-linear function in the forward pass. In the meantime, the emergence of the temporal-batch normalization [10] and residual gradient paths enables stable SNN training with deep models. In addition to the gradient approximation of the spiking function itself, recent work also investigates the gradient propagation along the temporal domain [11], where the spatial-temporal computation of SNN can be considered as a special version of a recurrent neural network (RNN) [11], [12]. The development of SNN training enables spike-based computer vision with large and deep models [13] on large-scale datasets (e.g., ImageNet).

b) SNN Compression: Motivated by the nature of SNN with binary spikes, prior SNN works have investigated low-precision quantization. Different from the static weight pruning in DNNs, sparsity in SNN can be explored in the weights and time domain. [3] investigated the lottery ticket hypothesis to SNNs, which explores the winning ticket in both weights and temporal steps for the computation skipping. [14] jointly compresses the low-precision weights down to 5-bit and sparsify the SNN with temporal pruning during training. Since SNN introduces the temporal dimension during inference, besides quantizing the weights [2], [5], considering the quantization of membrane potential is important for memory-efficient SNNs (Figure 1). Prior works [4], [15] have employed fixed-point representation for the membrane potential. However, none of these methods are verified against large-scale datasets or achieved the ultra-low-precision ($\leq$ 4-bit) for membrane potential.

III. Preliminary of SNN

a) Spiking Neurons: With the typical Leaky Integrate-and-Fire (LIF) neurons, the membrane potential is iteratively

accumulated through time while the binary spikes are generated after comparing the membrane potential with a predefined threshold.

$$u_t = (1 - \frac{dt}{\tau})u_{t-1} + \frac{dt}{\tau}I(t), \qquad S_t = \theta(u_t - V_{th}) \quad (1)$$

However, the spike function θ in LIF is non-differentiable. The infinite gradient of θ can be approximated by gradient surrogation with non-linear functions [7], [12].

b) Post-spike High-precision Membrane Potential: In general hardware implementation of SNN, the membrane potential of non-spiking pixels is offloaded to the memory cells and will be fetched back for the membrane accumulation of the next time step.

$$\hat{u}_t^l = u_t^l(1 - S_t^l) \quad (2)$$

In conventional SNN algorithm design, the membrane potential u_t will be saved in full precision with the size of $N \times C \times H \times W$. This iterative update and saving of membrane potentials, which necessitates high-precision values uniformly applied to each pixel, results in frequent memory access in SNNs. In addition, the low firing rate in deep SNNs further elevates memory demand due to the large-scale local storage of post-spike high-precision membrane potential.

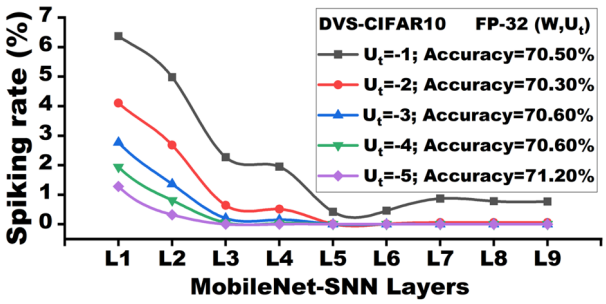
IV. Proposed Method

To achieve maximum hardware awareness with high performance, we propose IM-SNN with ultra-low precision integer-only-quantization of membrane potential and low-precision weights. The proposed IM-SNN addresses all the challenges to the low-precision representation of membrane in SNN and achieves maximum hardware efficiency.

A. Membrane Potential Clipping

We perform a two-step analysis to evaluate the robustness of SNN with respect to different quantization boundaries. Starting from the first time step of the SNN inference, the membrane potentials are stored in memory and get fetched for accumulation in the next time step. We define the negative clipping boundary as c, and the entire quantization range becomes $[c, \gamma]$, where γ is the maximum membrane potential after spiking. Naturally, $0 \leq \gamma < V_{th}$. The mechanism of "accumulate-and-fire" of SNN makes each membrane potential pixel possible to fire during the consecutive time steps. However, the relationship between membrane potential value and spiking activity is non-observable during inference. Let's assume that $\mathbf{\Gamma}_u$ represents the membrane pixels that are below the clipping threshold c. Naïvely unifying all the $\mathbf{\Gamma}_u$ values to the clipping threshold c will change the spiking rate of the next time step, and also the final output of the layer. To quantify such impact, we investigate the robustness of SNN with respect to different clipping thresholds c with the following two perspectives:

- **Step 1:** We first quantify the impact of the membrane clipping by analyzing the firing rate of pixels $\mathbf{\Gamma}_u$.
- **Step 2:** We evaluate the robustness and accuracy of the SNN model with different clipping thresholds.



Layer	1	2	3	4	5	6	7	8	9	Accuracy
Min Ut	-21.07	-10.56	-5.02	-5.65	-3.71	-4.33	-4.65	-5.26	-4.66	71.70 %

Fig. 2: Analysis of spiking activity with different membrane potential clipping boundaries.

Given the input sample X of the size of $N \times T \times C \times H \times W$, with total time steps = T, we first investigate the distribution of the post-spike membrane potential $\hat{u}_t^l$ at the very first time step $t = 1$. Mathematically, the pixels $\mathbf{\Gamma}_u$ are defined as:

$$\mathbf{\Gamma}_u = \{\hat{u}_{t=1}^l \mid \hat{u}_{t=1}^l < c\} \quad (3)$$

For the remaining $t \in [2, \dots T]$ time steps, the firing rate r_s of $\mathbf{\Gamma}_u$ can be computed as:

$$r_s = \frac{1}{T}\sum_t \frac{|\mathcal{M}_t|}{C \times H \times W} \quad \text{where } |M_t| = \mathbb{1}_{\mathbf{\Gamma}_u}(u^l) \wedge S_t^l \quad (4)$$

Where $\mathbb{1}$ is the indicator function and it returns the binary flag that represents whether the pixel u is unfired with a small magnitude. $\wedge$ is the "AND" logic, S_t^l is the binary spikes of layer l at time step t, and $|M|$ returns the number of non-zero elements. Therefore, r_s in Eq. (4) characterizes **how many membrane potential pixels are initially ($t = 1$) and inactive (less than c) but spike in the future t.**

Based on the theoretical setup above, we sweep over different clipping boundaries c from -5.0 to -1.0. For each value of c, we compute the average firing rate r_s across each $t \in [2 \dots T]$, and record the layer-wise firing rate (percentage). Figure 2 shows the layer-wise firing rate of a 9-layer lightweight MobileNet-V1 SNN, pre-trained on the DVS-CIFAR10 dataset with 30 total time steps. Compared to the widely-used ResNet-19, the lightweight MobileNet exhibits higher sensitivity to quantization [16], which provides an accurate insight into the clipping distortion.

Let's assume the minimal membrane potential (Min U_t) of each layer is U_{min}. As shown in Figure 2, the early layers exhibit a high magnitude of U_{min} (e.g., -21.07 for the first layer) in the floating point baseline. However, only 6.35% of membrane pixels U_t fire in the future time steps with all the accumulations, where $U_{\text{min}} < U_t < -1$. In other words, the remaining unfired 93.6% of membrane pixels create a "silent region", where the membrane pixels have **zero contribution** to future time steps and the next layer. Although U_{min} increases in the latter layers, the *silent region* in the negative membrane potential further extends for the next time

TABLE I: Clipping robustness of IM-SNN.

Negative Clips	FP	-5.0	-4.0	-3.0	-2.0	-1.0
CIFAR10-Accuracy	94.53	94.33	94.21	94.34	94.22	94.09

steps. As a result, clipping distortions have a negligible impact on the majority of the membrane potential, even with the most aggressive clipping (c=-1.0). Therefore, we have the following observation:

Observation: *If the membrane potentials are negative enough, their impact on spikes is minimal.*

We further prove the membrane clipping robustness by evaluating its impact on the accuracy during inference with no fine-tuning. As shown in Figure 2, starting from the pretrained full-precision model with 71.70% inference accuracy, when the membrane potential is clipped to $c = -1.0$, the inference accuracy can still be maintained at 70.50%. We extend the verification of this observation on the CIFAR-10 dataset with ResNet-19 and validate the negligible accuracy drop of 0.44% from the baseline by sweeping the clipping boundary from -5.0 to -1.0, as shown in Table I. The minimal accuracy degradation allows us to interpret the **Observation** into: *If the membrane potentials are negative enough, their impact on accuracy is minimum,* ***which implies the robustness of membrane potential to low-precision representation.***

B. Membrane Potential Quantization Algorithm

With the proven robustness of SNN using quantized membrane potential, recovering accuracy degradation becomes a critical task. In this paper, we propose a novel quantization algorithm designed for SNN training with low-precision membrane potential. Motivated by the *dequantization challenge* in Section I, the proposed method quantizes the membrane potential without introducing dequantization scaling [6], leading to high hardware awareness and efficiency.

Unlike prior works [15] that incorporated high-precision *scaling* for quantization, we design an integer-only quantization scheme to assign the membrane potential to the nearest integer level directly. Different from [4], [5] which uses high-precision *fixed point integer* on membrane potential, our method compresses the membrane potential precision down to **ternary**, as follows:

$$Q(u_t) = \begin{cases} -1 & \text{if} \quad u^t \leq 0 \\ 0 & \text{if} \quad u^t \leq 0.5 \\ +1 & \text{if} \quad u^t \geq 0.5 \end{cases} \tag{5}$$

The classic non-differentiable quantization operation hinders the backward propagation in both spatial and temporal directions, which can be factorized as:

$$\frac{\partial L}{\partial S_t} = \underbrace{\frac{\partial L}{\partial S_t^{l+1}} \frac{\partial S_t^{l+1}}{\partial S_t}}_{\text{w.r.t Layer}} + \underbrace{\frac{\partial L}{\partial S_{t+1}^{l}} \frac{\partial S_{t+1}^{l}}{\partial S_t}}_{\text{w.r.t time step}} \tag{6}$$

Where

$$\frac{\partial S_{t+1}^{l}}{\partial S_t} = \frac{\partial S_{t+1}^{l}}{\partial u_{t+1}^{l}} \frac{\partial u_{t+1}^{l}}{\partial u_t^{l}} \frac{\partial u_t^{l}}{\partial S_t} \tag{7}$$

Algorithm 1: PyTorch-style pseudocode for the membrane potential quantization algorithm

```
# T: total time steps of each input sample
# y: convolutional output
# c: clipping threshold of membrane
 potential
# lv: target quantization levels
# d: decay factor of membrane potential
 update
# mem: membrane potential of each time
 step
# thre: spiking threshold
def qfunc(mem, lv):
   mq = clamp(mem.view(-1), c, max(mem))
   idx = (mq.unsqueeze(0) -
    lv.unsqueeze(1)).abs().min(dim=0)[1] #
    distance and indexes

   out = lv[idx].reshape(mem.shape) #
    assign to nearest level
   return out

def forward(z):
   spikes = [] for t in T:
      mem = mem * d + y[:, t, ...] #
       membrane potential update
      spike = fire(mem - thre) # spike
      mem = (1 - spike) * mem # reset
      mem = qfunc(mem) # quantize membrane
       potential
      spikes.append(spike)
   return spikes
```

With our proposed quantization scheme, the membrane potential at each time step is updated with low precision:

$$u_{t+1}^{l} = \tau \times Q(u_t^l) + y_t^l \tag{8}$$

TABLE II: Different SGS functions and accuracy results on DVS-CIFAR10 dataset.

Architecture	Epochs	T	SGS	Accuracy (%)
VGG-9	200	30	ArcTan	77.81
VGG-9	200	30	Triangle	70.83
VGG-9	200	30	Piece-wise	78.81
VGG-9	200	30	**Sigmoid**	**80.04**

The temporal gradient $\partial u_{t+1}^{l} / \partial u_t^{l}$ is inaccessible due to the quantization function $Q(\cdot)$ which outputs integer levels only. To resolve this issue, we propose **S**tacked **G**radient **S**urrogation (SGS), which approximates the temporal gradient using the sigmoid function during the backward propagation to overcome the non-differentiability of quantization. The choice of Sigmoid function as SGS is empirical and the performance comparison of different surrogation functions is summarized in Table II. Formally, we define SGS-based Q^* as follows:

$$Q^*(u_t) = \sum_{k=1}^{K} T \frac{1}{1+e^{-T(u_{qt}-s_k)}}(1 - \frac{1}{1+e^{-T(u_{qt}-s_k)}}) \quad (9)$$

$$\text{and } s_k = (k_i + k_{i+1})/2 \quad (10)$$

Where T, k, and s_k represent the smoothness, quantization interval, and shift of each surrogation term, respectively. With SGS, we have:

$$\partial u_{t+1}^l / \partial u_t^l = \tau Q^*(u_t) \quad (11)$$

Combining Eq. (6)-(11), we formulate a smooth gradient propagation flow for SNN training with quantized membrane. Algorithm 1 describes the proposed quantization algorithm during the forward pass of training. Where `lv` represents the target quantization levels of membrane potential, which is defined by the precision of representation. For instance, `lv`=[-1.0, 0.0, 1.0] represents the ternary quantization, and the resultant membrane potential will be offloaded to the hardware memory with 1.5-bit representation.

C. Sparsified Low Precision Membrane

In addition to the ternarized membrane potential, the enhanced robustness of the SNN model exhibits strong resilience, which further allows "membrane pruning". Specifically, we directly de-active all the negative pixels on top of the ternarized membrane. Mathematically, we **disable** the spikes that correspond to the negative membrane potential, which is indexed by U_{mask}:

$$U_{mask} = \texttt{Bool}(Q(u_t) < 0) \quad (12)$$

$$S_t = S_t \cdot (1 - U_{mask}) \quad (13)$$

In particular, the negative membrane pixels are "pruned" from the spiking process, which is also skipped during the subsequent membrane potential updates. To evaluate this, we first perform an ablation study to estimate the layer-wise negative membrane potential and firing rate statistics using ResNet-19 on the CIFAR-100 dataset. Figure 3 demonstrates a very high percentage around $> 40\%$ of "-1" membrane pixels with below 10% firing rate across most of the ResNet-19-SNN layers. Such a high percentage of the negative membrane potential with the low firing rate occupies additional memory. This motivates us to skip the computation of all the negative "-1" integer membrane pixels completely.

$$U_{t+1} = U_t \odot U_{mask} \quad (14)$$

In other words, the membrane potential of the proposed IM-SNN can be further pruned by sparsifying the quantized ternary membrane, and the resultant membrane pixel values become binary before and after the spike operation. With the proposed IM-SNN offering a sparsified low-precision membrane, the resultant model shows high compute and memory efficiency with the pruned ternary membrane.

As shown in Table III, directly silencing all the "-1" of the membrane shows around 50% additional memory reduction with minimal accuracy degradation of $<0.97\%$ across different datasets, **without any fine-tuning**. The minimum accuracy degradation implies the improved robustness empowered by the IM-SNN training. Furthermore, the sparsified ternary membrane reduces the number of operations by $>57\%$ from the baseline by reducing 27.52 MOPs to 15.72 MOPs for the ResNet-19 model on the CIFAR-100 dataset and 18.43 MOPs to 7.74 MOPs for the VGG-9 model on DVS-CIFAR10 and N-Caltech datasets.

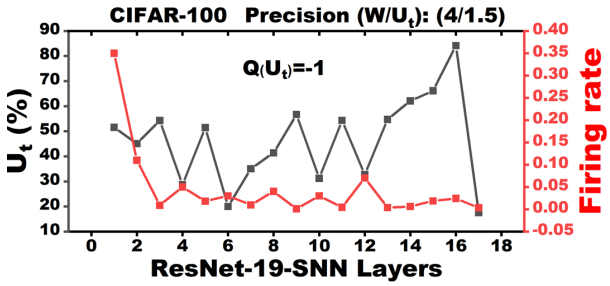


Fig. 3: Analysis of layer-wise negative membrane potential ($U_t(\%)$) and its firing rate for dynamic pruning.

TABLE III: IM-SNN performance with pruned ternary membrane potential on different datasets.

Architecture	Dataset	Masked Ternary Mem.	T	Membrane Memory (KB)	Accuracy
ResNet-19	CIFAR-100	No	2	250	71.87
ResNet-19	CIFAR-100	Yes	2	125	71.30
VGG-9	DVS-CIFAR-10	No	2	230	78.13
VGG-9	DVS-CIFAR-10	Yes	2	115	77.16
VGG-9	N-Caltech	No	2	230	80.04
VGG-9	N-Caltech	Yes	2	115	79.14

D. Weight Quantization in SNN Training

On top of membrane potential quantization, we incorporate low-precision weights to reduce memory occupancy. To train IM-SNN for low-precision inference, we adopt and modify the Power of Two (PoT) quantizer [17] to compress the layer-wise weights of the IM-SNN architecture down to 4-bit and 2-bit precision.

Binary input spikes of each layer convolving the low precision weights can be formulated as approximated computing or look-up tables, which further enhance the hardware efficiency in practice. In the end, we compute the total memory of IM-SNN and develop a total memory vs. accuracy comparison with the current SoTA SNN works.

V. Experiments and Results

We validate the proposed IM-SNN algorithm with both event-based and static image computer vision datasets. For event-based datasets, we train IM-SNN on DVS-CIFAR10 [18], N-Cars [19], N-Caltech101 [20] and Prophesee Automotive Gen1 [21] datasets. Further, we use CIFAR-10, CIFAR-100, ImageNet-100, and ImageNet-1k datasets for IM-SNN training and inference on static image datasets.

TABLE IV: Model architectures for IM-SNN training. “C3”, “DW”, “MP2” , “AP2” and “FC” represent 3×3 convolution layer, 2×2 max-pooling, 2×2 average-pooling, and fully connected layer, respectively.

Model	Architecture
MobileNet-Light	32C3-64DW-64DW-AP2-128DW- 128DW-AP2-256DW-AP2-FC256-FC10
VGG-7	32C3-32C3-AP2-64C3-64C3-AP2- 128C3-128C3-AP2-256C3-256C3-AP2-FC10
VGG-9	64C3-128C3-AP2-256C3-256C3-AP2- 512C3-512C3-512C3-512C3-AP2-1024C3-AP2-FC10
Custom-Yolo-V2	32C3-MP2-64C3-MP2-128C3-64C1-128C3-MP2- 256C3-128C1-256C3-MP2-512C3-256C1-512C3- 256C1-MP2-1024C3-512C1-1024C3-AP2-FC512-FC576
ResNet-19	64C3-128C3-128C3-128C3-128C3-128C3-128C3-128C3- 256C3-256C3-256C3-256C3-256C3-256C3-256C3-256C3- 256C3-512C3-512C3-512C3-512C3-AP2-FC256-FC10
ResNet-34	64C3-128C3-128C3-128C3-128C3-128C3-128C3-128C3- 256C3-256C3-256C3-256C3-256C3-256C3-256C3-256C3- 256C3-512C3-512C3-512C3-512C3-512C3-512C3-512C3- 512C3-512C3-512C3-512C3-512C3-512C3-AP2-FC256-FC10

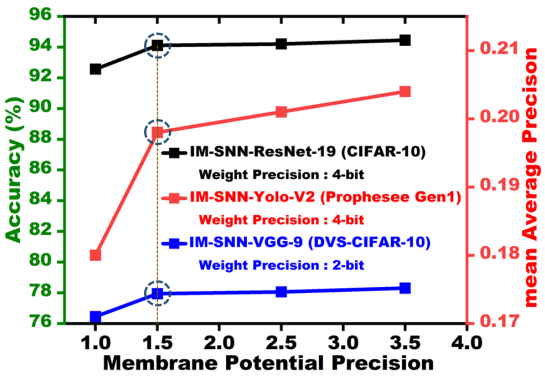


Fig. 4: Membrane potential precision vs. accuracy for CIFAR-10 and DVS-CIFAR10 datasets and membrane potential vs. mAP for Prophesee Gen1 object detection dataset. Note that ‘1.0’ membrane potential precision refers to the pruned ternary membrane presented in Section IV-C.

A. Experimental Setup

The proposed IM-SNN is evaluated on various datasets and a wide spectrum of model architectures including VGG-9, ResNet-19, ResNet-34, and Custom-Yolo-v2 presented in Table IV. In particular, for the static image, IM-SNN is employed with ResNet-19, ResNet-34, and Spikefomer for CIFAR-10, CIFAR-100, ImageNet-100, and ImageNet-1K classification. However, for DVS datasets, IM-SNN is implemented with VGG-9, Spikeformer, and custom-Yolo-V2 architectures for both classification and object detection tasks.

a) Data preprocessing: event to tensor conversion: The open-sourced DVS datasets are in the shape of an indexed bit stream where each chunk represents the axis information, spike polarity, time-step information, and addresses. At the data preprocessing stage, we extract spike polarity and time-step information from the bit streams and convert them to 48×48 binary frames. We sample over different time steps and formulate event tensors of 5-D shape [Batch, Time, Channels, Height, Width]. For static image data including CIFAR-10, CIFAR-100, and ImageNet, we use the 8-bit static images of shape [N, T, C, H, W] for the training and inference [22]. We repeat the static RGB frames by T times to introduce the temporal domain of the input.

On the other hand, we convert Prophesee Gen1 events to binary histograms by sampling over all the time steps. Then the generated binary histograms are synchronized with artificial ground truth from [23]. Finally, the events and annotations are translated to the tensors of shape [Batch, Time, Channels, Height, Width] and [Batch, Number of boxes, Bounding box] respectively. Unlike prior works [13], we do not use data augmentation for performance improvements.

b) IM-SNN training setup: We train the proposed algorithm with different architectures on 4 × A6000 GPUs with CUDA version 12.1 and Pytorch 2.0 [27]. All the models are trained using AdaM optimizer with the initial learning rate value of 0.001. The batch size is set to 128 for static image data and 32 for DVS image data. We shrink the resolution of DVS input images to 48×48 to optimize the training on available GPUs. To improve learnability, we optimize the learning rate with the Cosine annealing scheduler.

c) Choice of low-precision for membrane potential: With the proposed IM-SNN, we sweep across the different quantization levels for the membrane potential and weights to achieve high performance with maximum memory reduction. We optimally quantize the membrane potential to ternary levels $[-1.0, 0.0, 1.0]$ for each time step. The choice of ternary levels is based on the performance vs memory efficiency trade-off for both classification and object detection datasets, presented in Figure 4. Regarding low-precision of weights, we implement 8-bit, 4-bit, and 2-bit weight precision using the PoT quantization scheme [17] for IM-SNN training.

B. Evaluation Metric

We evaluate the IM-SNN based on memory reduction, robustness, and accuracy. We compute the total memory (MB) acquisition of each IM-SNN-based architecture for all the datasets. The total memory consists of weight, membrane potential, and convolutional output memory:

$$M_t = w_p \times N_w + u_p \times N_f + u_s \times N_f \tag{15}$$

$$\text{Where} \quad N_w = \sum_{l=1}^{\ell} w^l \tag{16}$$

$$\text{And} \quad N_f = \sum_{l=1}^{\ell} (C \times W \times H)^l \tag{17}$$

M_t is the total memory, w_p, u_p and u_s represent the weight, membrane potential and spike precision (0,1) respectively. Further, N_w, and N_f are total network weights and spikes.

C. Results and Discussion

a) RGB/Static image classification: IM-SNN is evaluated on both DVS and static computer vision datasets. For static image datasets, we compare the performance of IM-SNN with

TABLE V: Experimental results of IM-SNN on static image i-e. CIFAR-10, CIFAR-100, and ImageNet-100 datasets. Except "This work and MINT", all SoTA prior works in the table have used 32-bit weight precision and membrane potential.

Dataset	Method	Architecture	Weight Precision	U_{mem} Precision	Weight Memory (MB)	U_{mem} Memory (MB)	Total Memory (MB)	T	Top-1 Accuracy
CIFAR-10	tdBN [10]	ResNet-19	32-bit	32-bit	49.94	5.5	60.94	6	93.16%
	DSR [12]	ResNet-18	32-bit	32-bit	44.72	5.33	55.38	6	91.89%
	Dspike [1]	ResNet-18	32-bit	32-bit	44.72	5.33	55.38	6	94.25%
	Spikeformer [13]	Spikformer-4-256	32-bit	32-bit	38.20	NA	38.20	4	95.51%
	MINT [2]	ResNet-19	8-bit	8-bit	12.48	1.37	15.22	4	91.36%
	MINT [2]	ResNet-19	4-bit	4-bit	6.24	0.69	8.30	4	91.45%
	MINT [2]	ResNet-19	2-bit	2-bit	3.12	0.35	4.84	4	90.79%
	This work	**ResNet-19**	**32-bit**	**32-bit**	**49.94**	**5.5**	**60.94**	**6**	**94.56%**
	This work	**ResNet-19**	**4-bit**	**1.5-bit**	**6.24**	**0.25**	**7.49**	**2**	**94.11% (-0.45)**
	This work	**ResNet-19**	**2-bit**	**1.5-bit**	**3.12**	**0.25**	**3.87**	**2**	**92.89% (-1.57)**
	This work	**Spikformer-4-256**	**8-bit**	**1.5-bit**	**9.62**	**0.25**	**15.26**	**4**	**94.99% (-0.52)**
CIFAR-100	DSR [12]	ResNet-18	32-bit	32-bit	44.72	5.33	55.38	6	68.33%
	TET [8]	ResNet-19	32-bit	32-bit	49.94	5.33	60.94	2	72.87%
	This work	ResNet-19	**32-bit**	**32-bit**	**49.94**	**5.5**	**60.94**	**2**	**72.78%**
	This work	**ResNet-19**	**4-bit**	**1.5-bit**	**6.24**	**0.25**	**7.49**	**2**	**71.87% (-0.91)**
ImageNet-100	TET [8]	ResNet-34	32-bit	32-bit	87.19	11.03	109.25	2	74.76%
	This work	**ResNet-34**	**8-bit**	**1.5-bit**	**21.27**	**0.51**	**23.83**	**2**	**74.42% (-0.34)**
ImageNet-1k	tdBN [10]	ResNet-34	32-bit	32-bit	87.19	11.03	109.25	2	63.72%
	TET [8]	ResNet-34	32-bit	32-bit	87.19	11.03	109.25	2	68.00%
	Dspike [1]	ResNet-34	32-bit	32-bit	87.19	11.03	109.25	2	68.46%
	This work	**ResNet-34**	**8-bit**	**1.5-bit**	**21.27**	**0.51**	**23.83**	**2**	**67.82% (-0.64)**

TABLE VI: Experimental results of IM-SNN on DVS datasets i-e. DVS-CIFAR10 and N-Caltech. Except "This work", all SoTA prior works in the table have used 32-bit weight and membrane potential precision.

Dataset	Method	Architecture	Weight Precision	U_{mem} Precision	Weight Memory (MB)	U_{mem} Memory (MB)	Total Memory (MB)	T	Top-1 Accuracy
DVS-CIFAR10	tdBN [10]	ResNet-19	32-bit	32-bit	49.94	12.09	74.20	10	67.80%
	TET [8]	VGG-Like	32-bit	32-bit	40.65	3.68	48.01	10	77.33%
	DSR [12]	VGG-11	32-bit	32-bit	70.43	9.7	89.7	30	75.70%
	Dspike [1]	ResNet-18	32-bit	32-bit	44.72	11.7	68.12	10	75.45%
	Spikeformer [13]	Spikformer-16-256	32-bit	32-bit	38.5	NA	38.5	16	80.90%
	This work	**VGG-9**	**32-bit**	**32-bit**	**41.12**	**3.68**	**48.58**	**10**	**78.45%**
	This work	**VGG-9**	**2-bit**	**1.5-bit**	**2.57**	**0.23**	**3.75**	**10**	**77.94% (-0.51)**
N-Caltech	YOLE [24]	VGG7-Like	32-bit	32-bit	42.69	2.01	46.71	1	70.02%
	EST [25]	ResNet-34	32-bit	32-bit	88.39	43.4	175.19	1	78.70%
	Asynet [26]	VGG-13	32-bit	32-bit	22.32	4.09	30.50	1	76.10%
	This work	**VGG-9**	**32-bit**	**32-bit**	**41.12**	**3.68**	**48.58**	**10**	**80.45%**
	This work	**VGG-9**	**2-bit**	**1.5-bit**	**2.57**	**0.23**	**3.75**	**10**	**79.45% (-1.0)**

existing full-precision and low-precision [2] SoTA works. We compare the proposed algorithm with existing SoTA in the context of total memory, time step, and top-1 accuracy. As shown in Table V, we achieve up to 8.13× reduction in memory consumption with minimal accuracy degradation of 0.45% and 0.91% from SoTA SNN baseline for CIFAR-10 and CIFAR-100. Furthermore, IM-SNN achieves 4.58× memory reduction with 0.34% and 0.64% accuracy drop from the SNN baseline on the ImageNet-100 and Imagenet-1K datasets.

In addition, compared to the recent MINT-SNN which incorporates low-precision membrane potential from 8-bit to 2-bit, the proposed IM-SNN with 2-bit precision of weights and 1.5-bit precision of membrane achieves 2.1% higher accuracy with 1.23× memory reduction, as shown in Table V.

b) DVS data classification: For the DVS dataset, we train IM-SNN-VGG9 and IM-SNN-Spikeformer for direct comparison of the proposed algorithm with existing works, and results are shown in Table VI. Using IM-SNN-VGG9, we attain up to 13× memory reduction with an accuracy drop of 0.51% and 1% from the full-precision baseline for DVS-CIFAR-10 and N-Caltech datasets. Compared with existing SoTA, the proposed IM-SNN scheme surpasses the cutting-edge SNN with 0.59% and 0.70% accuracy improvement for DVS-CIFAR-10 and N-Caltech datasets respectively and reduces the total memory consumption by up to 13×.

c) Object Detection with IM-SNN: We further demonstrate the performance of the proposed IM-SNN on a large-scale Automotive Prohesee Gen1 dataset [21]. The DVS events are converted to binary frames synchronized with their actual ground truths. To avoid the gradient vanishing in IM-SNN, we customize the Yolo-V2 model by skipping one convolution block from the original architecture. We train IM-SNN-Custom-Yolo-V2 architecture with ternary-level representation for membrane potential $[-1.0, 0.0, 1.0]$ and 4-bit precision for weights. As shown in Table VII, the proposed IM-SNN reduces memory utilization by 7.08× in comparison to the full-precision baseline with a decrement of 0.042 in the mAP. Furthermore, compared to SNN-based object detection SoTA work VGG11-SSD [28], IM-SNN-Custom-Yolo-V2 achieves 0.011 higher mAP with 9.58× reduced memory footprint.

VI. Conclusion

In this paper, we propose IM-SNN, a novel SNN algorithm that enables integer-only SNN with largely reduced memory

TABLE VII: Experimental results of the IM-SNN-Custom-Yolov2 on Prophesee Automotive Gen1 dataset.

Dataset	Method	SNN	Weight Precision	U_{mem} Precision	Weight Memory(MB)	U_{mem} Memory(MB)	Total Memory(MB)	mAP
Asynet [26]	FB-Dense	No	32	-	532.00	-	532.00	0.145
MatrixLSTM [9]	ResNet-19	No	32	-	260.00	-	260.00	0.300
RED [23]	RetinaNet	No	32	-	96.00	-	96.00	0.410
VGG-11+SSD [28]	**VGG+SSD-SNN**	Yes	32	32	141.61	80.39	302.39	0.187
This work	**Custom-YoloV2-SNN**	**Yes**	**32**	**32**	**103.46**	**59.87**	**223.20**	**0.240**
This work	**Custom-YoloV2-SNN**	**Yes**	**4**	**1.5**	**12.93**	**3.76**	**31.57**	**0.198**

consumption. The proposed algorithm successfully compresses the membrane potential down to ternary representation, achieving up to 13× memory footprint reduction, while maintaining the high simplicity and high performance of SNN. Furthermore, IM-SNN shows strong robustness in dynamic membrane pruning. The sparsified membrane opens up the potential of on-device computation skipping during the inference. IM-SNN is evaluated on a comprehensive spectrum of computer vision tasks, including both static image classification and event-based object detection. The outstanding versatility makes the proposed IM-SNN a powerful solution for energy-efficient computer vision.

ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation under grant 2403723, and CoCoSys Center in JUMP 2.0, an SRC program sponsored by DARPA.

REFERENCES

[1] Y. Li, Y. Guo, S. Zhang, S. Deng, Y. Hai, and S. Gu, "Differentiable spike: Rethinking gradient-descent for training spiking neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[2] R. Yin, Y. Li, A. Moitra, and P. Panda, "Mint: Multiplier-less integer quantization for spiking neural networks," *arXiv preprint arXiv:2305.09850*, 2023.

[3] Y. Kim, Y. Li, H. Park, Y. Venkatesha, R. Yin, and P. Panda, "Exploring lottery ticket hypothesis in spiking neural networks," in *European Conference on Computer Vision (ECCV)*. Springer, 2022.

[4] R. V. W. Putra and M. Shafique, "Q-spinn: A framework for quantizing spiking neural networks," in *International Joint Conference on Neural Networks (IJCNN)*, 2021.

[5] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, 2018.

[6] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and Training of Neural Networks for Efficient Integer-arithmetic-only Inference," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[7] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-based Optimization to Spiking Neural Networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, 2019.

[8] S. Deng, Y. Li, S. Zhang, and S. Gu, "Temporal Efficient Training of Spiking Neural Network via Gradient Re-weighting," in *International Conference on Learning Representations (ICLR)*, 2021.

[9] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "A differentiable recurrent surface for asynchronous event-based data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*. Springer, 2020.

[10] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going Deeper with Directly-trained Larger Spiking Neural Networks," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, no. 12, 2021.

[11] G. Shen et al., "Backpropagation with Biologically Plausible Spatiotemporal Adjustment for Training Deep Spiking Neural Networks," *Patterns*, p. 100522, 2022.

[12] Q. Meng, M. Xiao, S. Yan, Y. Wang, Z. Lin, and Z.-Q. Luo, "Training High-Performance Low-Latency Spiking Neural Networks by Differentiation on Spike Representation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[13] Z. Zhou, Y. Zhu, C. He, Y. Wang, S. Yan, Y. Tian, and L. Yuan, "Spikformer: When spiking neural network meets transformer," in *International Conference on Learning Representations (ICLR)*, 2023.

[14] S. S. Chowdhury, I. Garg, and K. Roy, "Spatio-temporal pruning and quantization for low-latency spiking neural networks," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[15] R. V. W. Putra and M. Shafique, "tinysnn: Towards memory-and energy-efficient spiking neural networks," *arXiv preprint arXiv:2206.08656*, 2022.

[16] E. Park and S. Yoo, "Profit: A novel training method for sub-4-bit mobilenet models," in *European Conference on Computer Vision (ECCV)*. Springer, 2020.

[17] D. Przewlocka-Rus, S. S. Sarwar, H. E. Sumbul, Y. Li, and B. De Salvo, "Power-of-two quantization for low bitwidth and hardware compliant neural networks," *arXiv preprint arXiv:2203.05025*, 2022.

[18] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "Cifar10-DVS: An Event-stream Dataset for Object Classification," *Frontiers in Neuroscience*, vol. 11, 2017.

[19] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "HATS: Histograms of Averaged Time Surfaces for Robust Event-based Object Classification," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades," *Frontiers in Neuroscience*, vol. 9, 2015.

[21] P. De Tournemire, D. Nitti, E. Perot, D. Migliore, and A. Sironi, "A large scale event-based detection dataset for automotive," *arXiv preprint arXiv:2001.08499*, 2020.

[22] I. Garg, S. S. Chowdhury, and K. Roy, "Dct-snn: Using dct to distribute spatial information over time for low-latency spiking neural networks," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[23] E. Perot, P. De Tournemire, D. Nitti, J. Masci, and A. Sironi, "Learning to Detect Objects with a 1 Megapixel Event Camera," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[24] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Asynchronous convolutional networks for object detection in neuromorphic cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[25] D. Gehrig, A. Loquercio, K. G. Derpanis, and D. Scaramuzza, "End-to-end learning of representations for asynchronous event-based data," in *IEEE/CVF International Conference on Computer Vision (CVPR)*, 2019.

[26] N. Messikommer, D. Gehrig, A. Loquercio, and D. Scaramuzza, "Event-based Asynchronous Sparse Convolutional Networks," in *European Conference on Computer Vision (ECCV)*, 2020.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS) 32*, 2019.

[28] L. Cordone, B. Miramond, and P. Thierion, "Object detection with spiking neural networks on automotive event data," in *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022.