Spiking Neural Network with Learnable Threshold for Event-based Classification and Object Detection

Ahmed Hasssan
Cornell Tech, USA
ah2288@cornell.edu

Jian Meng
Cornell Tech, USA
jm2787@cornell.edu

Jae-sun Seo
Cornell Tech, USA
js3528@cornell.edu

2024 International Joint Conference on Neural Networks (IJCNN) | 979-8-3503-5931-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IJCNN60899.2024.10650320

***Abstract*—Spiking neural networks (SNNs) have received increasing attention due to their high biological plausibility and energy efficiency. The binary spike-based information propagation enables efficient sparse computation for event-based computer vision applications. However, most prior works use the heuristically selected fixed threshold for spiking neurons, which limits the dynamics of SNNs toward further optimizing the performance. In the meantime, the optimization space of the existing trainable spike neurons is often limited by various constraints. Motivated by this, this paper investigates the plausibility of freely optimizing the threshold during direct SNN training. Specifically, we propose LT-SNN, a novel SNN training algorithm with a self-adaptive learnable potential threshold to improve SNN performance. LT-SNN optimizes the layer-wise firing threshold throughout SNN training without any high-precision spike representation or learning constraints. Extensive experiments are performed across event-based and static computer vision datasets, including both image classification and object detection tasks. Equipped with high adaptiveness that fully captures the dynamics of SNNs, LT-SNN outperforms the recent state-of-the-art works. Furthermore, LT-SNN is compatible with SNN models based on both convolutional neural networks (CNN) and vision transformers (ViT).**

***Index Terms*—Spiking Neural Networks, Energy-efficient neural networks, Neuromorphic computing, Learnable threshold.**

I. Introduction

In the biological nervous system, cortical neurons process information by encoding spatial-temporal inputs into action potentials for spike generation. Inspired by that, spiking neural networks (SNNs) extract information from the spatial-temporal input, accumulate the membrane potential, and the resultant binary spikes (0 and 1) provide a sparse and succinct information representation. Such information propagation and computation procedure promotes SNN as an attractive AI solution with both biological plausibility and energy efficiency in comparison to the conventional artificial neural networks (ANNs) [1]. Furthermore, layer-wise processing with binary spikes elevates the computation efficiency, which benefits energy-constrained applications such as edge computing.

On the other hand, event-based camera or dynamic vision sensors (DVS) [2] have emerged as an attractive and feasible solution for computer vision applications. Compared to the conventional frame-based cameras, event cameras independently capture the absolute illumination changes of pixels, resulting in the asynchronous binary stream of events [3]. The captured event is characterized by binary pixels and temporal resolutions, leading to highly sparse and energy-efficient visual

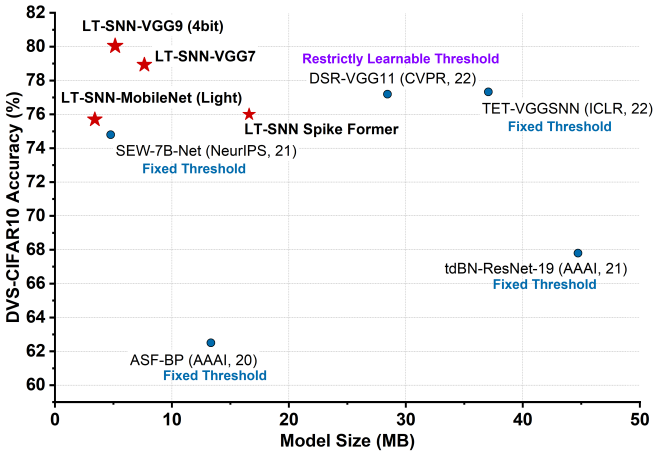

Fig. 1: DVS-CIFAR10 classification accuracy of different SNN training methods. The proposed LT-SNN training algorithm achieves SoTA accuracy with compact model.

representations. Such binarized spatial-temporal information naturally fits the computation mechanism of SNNs, bridging the gap between the efficient computer vision and neuromorphic computing.

To obtain SNN models, early research works relied on the ANN-to-SNN conversion [4], [5]. However, this approach requires additional training without achieving sufficiently high accuracy. To that end, direct training methods have been proposed to obtain an SNN model with one-time training. One of the major bottlenecks of direct SNN training is the non-differentiability of the spike generation. The infinite gradient of the spike function impedes the gradient propagation during the backward pass in training. Empowered by various surrogate gradient (SG) functions [6]–[8], the inaccessible gradient of the spike function is approximated and propagated during learning. However, the inaccurate approximation and heuristic SG selection hurt the training stability with deep models (e.g., ResNet [1]), which further motivated the temporal normalization method [9] and output regularization techniques [6], [10] to smooth the loss.

a) Adaptive threshold in biological neurons: As the major inspiration of deep learning, the intricate nervous system achieves remarkable performance with a high degree of dynamics. Recent works observed the location-dependent potential threshold [11] in nervous systems, implying the adaptive

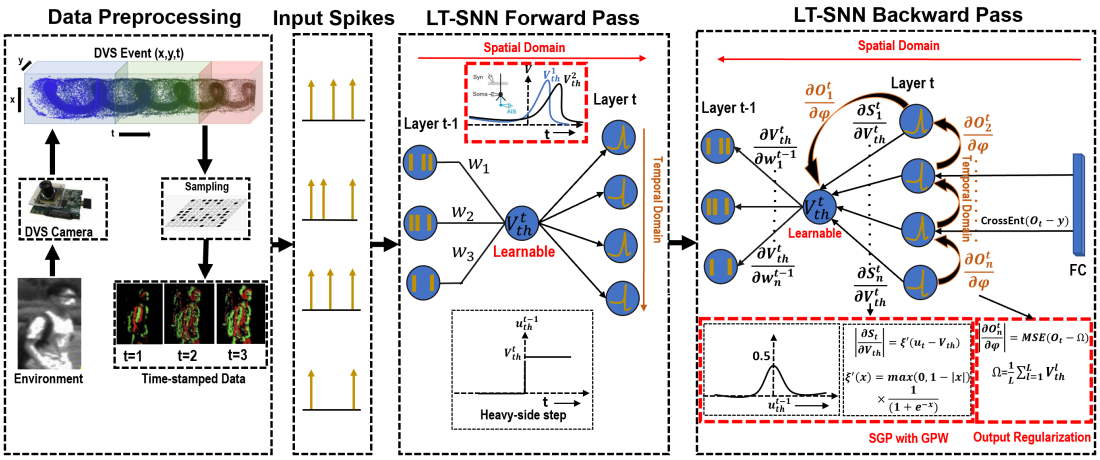

Fig. 2: Overview of the proposed LT-SNN algorithm.

firing procedure within the mechanism of spike generation. Further, intrinsic excitability at neuron level aligns with the dynamic spectrum of synaptic inputs they receive during the learning process, known as neuronal intrinsic plasticity [12], [13]. This significantly impact the biophysical characteristics of neurons at a cellular level, leading to a more nuanced understanding of neural functionality and learning mechanisms. These insights potentially link the dynamic spike threshold and the intrinsic excitability of neurons, implying that the spike thresholds could be considered a learnable parameter [14], [15].

b) Recent learnable dynamics in SNNs: Inspired by this, some recent works on SNN training introduced the learning dynamics into the spiking process, albeit to a limited degree. ParamLIF [16] optimized the membrane time constant throughout training, with the requirements of large-sized models. [17] presented the adaptability of the spike threshold in neurons as hyperparameter tuning. These hyperparameters are considered uniform across all neurons and are kept constant throughout the training process, without undergoing any adaptation. DSR [18] proposed threshold-associated spikes with a learnable potential threshold. However, the heuristic and deterministic high-precision ratio between the firing range and the potential threshold of DSR limits the adaptiveness of SNN. [19] implemented weight-threshold balancing to improve the SNN adaptability to the input data for enhanced firing rate in deep SNNs. However, considering the weights and potential threshold same landscape makes the learning sub-optimal.

Given the fact that the current SoTA performance [6] is mainly achieved with fixed potential threshold, the under-explored adaptiveness of the membrane potential threshold limits the learnability of the current generation SNN. The limitations of all prior works motivate us to investigate the question: *How can we optimize the potential threshold during SNN training with maximum adaptability, high stability and superior accuracy?*

To answer this question, we propose *LT-SNN*, a novel self-adaptive **SNN** training algorithm with **L**earnable **T**hreshold. Starting the training from scratch, LT-SNN fully optimizes the potential threshold without introducing additional high-precision spikes or firing constraints. We consider the weights and membrane threshold learning as two separate landscapes and optimize them individually in the backward propagation. To achieve highly stable training, we propose a simple-yet-effective technique, called Separate Gradient Path (SGP) for membrane potential optimization.

The main contribution and highlights of proposed LT-SNNs are summarized below:

- This paper proposes the layer-wise learnable threshold for spike operation to implement the biological level adaptability in SNNs without introducing extra computations.
- Compared to prior works, the proposed LT-SNN algorithm fully unleashes the advantage of unconstrained layer-wise adaptive potential threshold leaning, leading to superior performance without extra computations.
- The proposed LT-SNN eliminates the gradient mismatch problem by introducing SGP for neuron threshold learning, separating it from conventional weights learning in directly trained SNNs.
- We validate LT-SNN on multiple event-based and static image-based computer vision datasets with various SNN architectures including those based on CNNs and ViTs. LT-SNN achieves the new SoTA performance with lightweight and quantized models, as shown in Figure 1.

II. Related Works

a) ANN-SNN conversion: Due to the non-differentiability of the spike function, early research works converted high-performance non-spiking ANN model into a spiking version [4], [20]. The major drawbacks of the conversion-based method are high computation latency (high number of time steps) and the additional efforts for the overall training. Several methods have been proposed to improve the latency of the converted model [5], but still did not reach the high accuracy of direct SNN training methods.

b) Direct SNN training: The expensive training effort and high latency of the conversion-based schemes promote direct SNN training to be an attractive solution. BNTT [21]

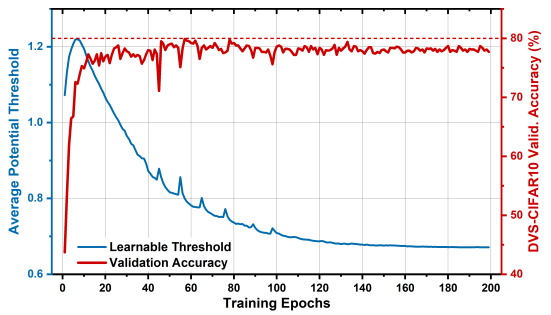

Fig. 3: Stable training process of the proposed LT-SNN algorithm with the extended training effort on DVS-CIFAR10 dataset.

treats the spatial-temporal computation of SNN as a special version of a recurrent neural network (RNN). The gradient of the spike function is approximated by surrogate gradient (SG) functions in backpropagation. Driven by accuracy and different model architectures, various SG functions have been proposed, including but not limited to rectangle function [8], arctangent [16], [22], and triangle functions [6]. However, the difference between the approximated gradient and the exact gradient largely limits the training stability of SNN, especially for large-sized models. Motivated by that, tdBN [9] introduces batch-temporal normalization for deep SNN training, and SEW-ResNet [22] directly passes the gradient via the designed residual architecture. In addition to the architecture design, various regularization techniques were presented to stabilize SNN training by rectification of the membrane potential distribution [10] and backpropagation with spatio-temporal adjustment [23].

c) SNN with Adaptive Neurons: Unlike the conventional ANN with simple activation functions (Sigmoid, ReLU), SNN training involves the parametrized neurons (e.g., LIF neurons) with spike functions. To avoid the heuristic parameter selection, prior works optimized the parameters of spike neurons via training. [16] introduces the learnable time constant for direct SNN training, but the employed large-sized SNN model and extensive training efforts (up to 1,024 epochs) are computationally expensive. Recently proposed DSR [18] optimizes the potential threshold during training by multiplying the binary output spike with the threshold value, where the relationship between the firing range and the threshold value is constrained by a deterministic ratio. However, optimizing the potential threshold value with such additional constraints limits the learnability of the SNN model. Furthermore, the threshold-dependent spikes introduce high-precision data into the subsequent layers, which deteriorates the hardware compatibility with expansive high-precision multiplication, since spikes are not binary anymore.

III. Background on Spiking Neural Networks

Inspired by the biological nervous system, spiking neural networks propagate the spatial-temporal information through the "integrate-and-fire" procedure.

$$\tau \frac{du_t}{dt} = -(u_t - u_{reset}) + I(t) \tag{1}$$

$$u_t = (1 - \frac{dt}{\tau})u_{t-1} + \frac{dt}{\tau} I(t) \tag{2}$$

where $u(t)$, u_{reset}, $I(t)$, and τ represent the membrane potential, reset potential, the synapse current, and the time constant respectively. Given the iteratively accumulated membrane potential, the binary spikes are generated by the step function:

$$S_t = \theta(u_t - V_{th}) = \begin{cases} 1 & \text{if } u_t \geq V_{th} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Where θ represents the Heaviside step function, and V_{th} represents the membrane potential threshold for spiking neurons. In the forward pass of SNN, the binary post-synaptic spikes are generated when the membrane potential exceeds the potential threshold V_{th}. With the spatial-temporal backpropagation (STBP) [8], the backward pass of SNN is characterized as:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial S_t} \frac{\partial S_t}{\partial u_t} \frac{\partial u_t}{\partial I_t} \frac{\partial I_t}{\partial W} \tag{4}$$

Due to the non-differentiable spike function θ, the directly-trained SNN incorporates the surrogate gradient function to approximate the intangible Dirac function. In this work, we choose the triangle function for gradient approximation:

$$\frac{\partial S_t}{\partial u_t} = \theta'(u_t - V_{th}) = \max(0, 1 - |u_t - V_{th}|) \tag{5}$$

TABLE I: Model architectures for LT-SNN training. "C3", "DW", "MP2" , "AP2" and "FC" represent 3×3 convolution layer, depth-wise separable block [24], 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	64C3-128C3-128C3-128C3-128C3-128C3-128C3-256C3-256C3-256C3-256C3-256C3-256C3-256C3-256C3-512C3-512C3-512C3-512C3-AP2-FC256-FC10
ResNet-26	64C3-128C3-128C3-128C3-128C3-128C3-128C3-256C3-256C3-256C3-256C3-256C3-256C3-256C3-256C3-512C3-512C3-512C3-512C3-512C3-512C3-512C3-512C3-512C3-512C3-AP2-FC256-FC10

IV. Towards Optimizing Potential Threshold

As described in Eq. (5), the non-differentiability between the output spike S and the membrane potential u can be alleviated by the surrogate gradient approximation [7]. With the same type of surrogation, the gradient of the potential threshold V_{th} can be naïvely approximated in a straight-through manner using vanilla surrogate gradient (SG):

$$\frac{\partial L}{\partial V_{th}} = -\frac{\partial L}{\partial S_t} \frac{\partial S_t}{\partial V_{th}} = -\frac{\partial L}{\partial S_t} \theta'(u_t - V_{th}) \tag{6}$$

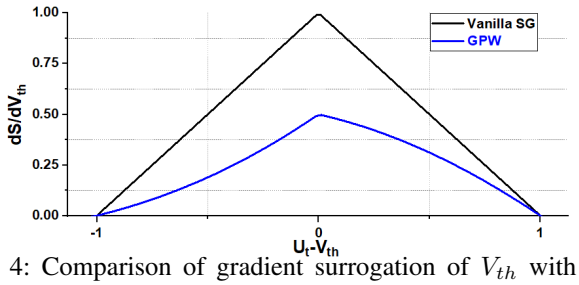

Fig. 4: Comparison of gradient surrogation of V_{th} with and without GPW

TABLE II: The performance of DSR [18] is largely impacted by the value of α. The proposed LT-SNN outperforms DSR with 2.24% higher accuracy on the DVS-CIFAR10 dataset.

Method	Model	Accuracy (%)	α	True Binary Spike
DSR [18]	VGG-11	77.27	0.3	✗
DSR [18]	VGG-11	75.29	1.0	✗
This work	VGG-11	**79.51**	-	✓

The straight-through approximation in Eq. (6) is based on an assumption that the surrogate gradient of $\frac{\partial S_t}{\partial u_t}$ and $\frac{\partial S_t}{\partial V_{th}}$ are transferable. However, such transferability remains unjustified, and the optimality of SG remains uninvestigated.

Different from the vanilla SG approach, the recent DSR scheme [18] computes the gradient of the potential threshold with the following threshold-based firing procedure:

$$S_t = V_{th} \times \theta(u_t - \alpha V_{th}) \tag{7}$$

Where the fixed parameter $\alpha \in [0, 1]$ controls the threshold with respect to the firing range $[0, V_{th}]$. In other words, regardless of the threshold value, the membrane potential has to be a deterministic portion of the total firing range $[0, V_{th}]$. However, such constraints in the training process largely limit the learnability of SNN. Based on the open-sourced implementation of DSR [18], we unleash the optimization constraints of the potential threshold by setting $\alpha = 1.0$. Please note that, keeping the threshold and spike as identical values is commonly adopted in prior works [6]. As shown in Table II, the freely-learned potential threshold ($\alpha = 1.0$) exhibits large accuracy degradation for the DVS-CIFAR10 dataset compared to the constrained learning ($\alpha = 0.3$), which implies the sub-optimal performance with constrained threshold optimization.

In the context of hardware computation, layer-wise varied spikes of DSR [18] require the high-precision data representation of the output activation. Storing and processing high-precision data magnifies the memory consumption for subsequent computations. Furthermore, unlike the conventional ANN with a one-time computation (e.g., convolution) in each layer, the high precision scaling of DSR lead to high precision computation at every single time step, which could lead to increased energy and hardware resource consumption. These inefficiencies raise the following question:

Question 1: *How to improve the performance of SNN with the freely-optimized potential threshold, without losing the hardware compatibility?*

TABLE III: Performance comparison on DVS-CIFAR10 dataset with different learning rates and initial threshold.

Architecture	lr-W	lr-V_{th}	Initial V_{th}	SGP	Top-1 Accuracy (%)
VGG-9	0.1	0.001	1.0	✗	20.40
VGG-9	0.001	0.0001	1.0	✗	78.35
VGG-9	0.001	0.0005	1.0	✗	78.60
VGG-9	0.001	0.0005	4.0	✗	77.40
VGG-9	**0.001**	**0.001**	**1.0**	✓	**80.07±0.12**

V. Proposed Method

To answer the above question, we propose *LT-SNN*, a novel SNN training algorithm that collectively achieves the potential threshold optimization, training stability, and hardware compatibility. LT-SNN optimizes the layer-wise potential threshold during direct SNN training, maximizing the training performance of SNNs without introducing any learning constraints. In the meantime, LT-SNN embraces the advantages of the adaptive potential threshold while maintaining the true binary spikes, bridging the gap between biological dynamics and practical AI applications. The forward pass of LT-SNN computes the membrane potential and generates spikes with LIF neurons, while the potential threshold learnability is added during the backward pass. A separate gradient path computes the gradient value to tune the layer-wise potential threshold. The complete flow of the proposed LT-SNN is illustrated in Figure 2.

A. Non-transferrability of vanilla SG

In vanilla SG approximation, a unified surrogate gradient function is used (for both weights and threshold). It provides training simplicity by introducing a naturally fitting learning mechanism in SNNs. However, we applied vanilla SG under different learning rate schemes using DVS-CIFAR10 and observed that the simplicity of training does not guarantee the optimality of learning. We infer from Table IV results that naïvely sharing the surrogate gradient function failed to outperform the fixed potential threshold counterpart, regardless of the learning rate schemes. The suboptimal performance of vanilla SG with different learning rates and initial threshold values presented in Table III implies the necessity of a dedicated optimization method designed for potential threshold.

TABLE IV: Training SNN with SG on DVS-CIAR10 dataset and VGG-9 architecture.

Method	Threshold	lr of weight	lr of V_{th}	Accuracy (%)
TET [6]	Fixed	0.001	-	77.33
Vanilla-SG-TET	Learnable	0.001	0.00025	76.50
Vanilla-SG-TET	Learnable	0.001	0.00001	77.20

B. Separate Gradient Path

The suboptimal performance of vanilla SG implies the incompatibility of using the unified gradient surrogation when we perform SNN training with the trainable threshold. Motivated by that, we separate the gradient computation of V_{th} and u_t by proposing a Separate Gradient Path (SGP). Specifically, we develop SGP using a Gradient Penalty Window (GPW), a

TABLE V: Comparison of different surrogate functions' performance as SGP for DVS-CIFAR10 dataset.

Model	Epochs	SGP	Converged	Accuracy (%)
VGG-9	200	ArcTan	✓	77.81
VGG-9	200	Triangle	✓	70.83
VGG-9	200	Piece-wise	✓	78.81
VGG-9	200	Sigmoid	✓	80.20

simple-yet-effective method that is dedicated to the gradient computation of the potential threshold. On top of the gradient approximation in Eq. (5), GPW is characterized as a non-linear function $\sigma(\cdot)$, which reshapes the surrogate gradient of the layer-wise potential threshold V_{th}. Mathematically, the GPW-aided separate gradient path is characterized as:

$$\frac{\partial S_t}{\partial u_t} = \theta'(u_t - V_{th}) = \max(0, 1 - |u_t - V_{th}|) \tag{8}$$

$$\frac{\partial S_t}{\partial V_{th}} = -\theta'(u_t - V_{th})\sigma(u_t - V_{th}) \tag{9}$$

$$\frac{\partial S_t}{\partial V_{th}} = -\max(0, 1 - |u_t - V_{th}|)\sigma(u_t - V_{th}) \tag{10}$$

In this work, we choose the Sigmoid function as the gradient penalty window for the potential threshold:

$$\sigma(u_t - V_{th}) = \frac{1}{1 + e^{-(u_t - V_{th})}} \tag{11}$$

The choice of Sigmoid function is empirical as it produces the best results among different surrogate functions that we have experimented. Performance comparison of different surrogate functions with LT-SNN is shown in Table V. For gradient computation of V_{th}, we accumulate the gradient computed in Eq. (9) to avoid the dimensionality mismatch:

$$|\frac{\partial L}{\partial V_{th}}| = \frac{\partial L}{\partial S_t}\frac{\partial S_t}{\partial V_{th}} = \frac{\partial L}{\partial S_t}\sum_{(N,C,H,W)} \left(1\{u_t \geq V_{th}\} \times \theta'(u_t - V_{th})\sigma(u_t - V_{th})\right) \tag{12}$$

Since the unfired neurons have no contribution to the final loss, the indicator function $1\{u_t \geq V_{th}\}$ filters the gradient with respect to the active neurons in the forward pass.

Additionally, LT-SNN achieves >78% validation accuracy within 50 training epochs, as depicted in Figure 3. Compared to the SoTA SNN training with fixed threshold [6], SGP embraces the advantage of the adaptive potential threshold learning and achieves superior accuracy. Furthermore, SGP preserves the true binary spikes (0 and 1) in the resultant model, maximizing the hardware compatibility without introducing any high-precision scaling for spike generation [18].

a) Rationality of SGP and GPW: As shown in Eq (9)-(10), the impact of the Sigmoid-based gradient penalty window (GPW) is two-fold: 1) The peak gradient value has been reduced by 0.5× after applying the GPW. 2) With the same input u_t, the magnitude of $\frac{\partial S_t}{\partial V_{th}}$ weighted higher when $u_t - V_{th} > 0$. Figure 4 shows the comparison between the gradient surrogation of V_{th} with and without GPW.

However, the experiment in Table IV implies the fact that the slowly-trained V_{th} cannot guarantee the optimality of learning. Reducing the magnitude of the gradient has limited improvement in threshold learning. On the other hand, training the threshold by highlighting the gradient w.r.t the fired neurons penalizes the distortion of the unfired pixels. Therefore, the essence of the proposed SGP+GPW is equivalent to the separate regularization between the fired and unfired neurons of SNN. Our experiments in Table IV and Table V prove the benefits of learning the V_{th} with active neurons. Our experimental results in the next section validate our findings with both CNN and vision transformers (ViT).

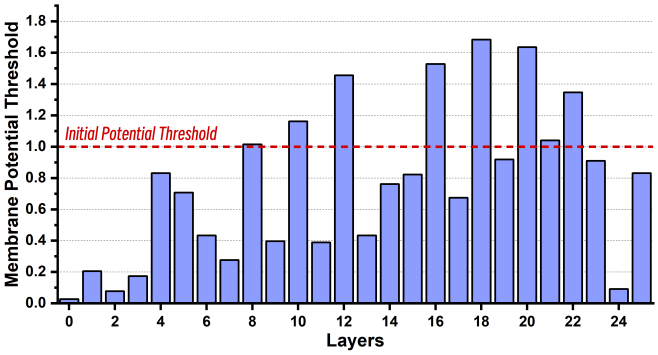

Fig. 5: Layer-wise adaptive potential threshold V_{th}^l of ResNet-26 trained by ImageNet-100 dataset with LT-SNN.

VI. Experimental Results

We validate the proposed LT-SNN algorithm with a wide range of event-based computer vision datasets, including DVS-CIFAR10 [29], N-Cars [30], N-Caltech101 [31] and Prophesee Automotive Gen1 [32]. Unlike prior works that only employ large-sized CNN models (e.g., VGG or ResNet models), the proposed algorithm is also validated on compact VGG models (2.4-7.1M parameters), light-weight MobileNet-V1 model (1.5M parameters) [24], as well as spike-based transformers [27]. In addition to the event-based computer vision datasets, we also validate the proposed LT-SNN algorithm with the conventional static image datasets, including CIFAR-10, CIFAR-100 ImageNet-100 and ImageNet-1k datasets. Table I summarizes the performance comparison of different models and datasets.

a) Training Setup: We train our proposed LT-SNN-based classification and object detection architectures using PyTorch [33] version 1.9.0 with CUDA version 11.1. Regarding hyperparameter selection, we use the Adam optimizer where the learning rate is set to 0.001. We computed TET [6] loss between the logits and the target labels. The regularization level β is set to 0.45 and 0.90 for both the full-precision and low-precision training of VGG and ResNet architectures, respectively.

b) SNN Training for Full-/Low-Precision Inference: We perform LT-SNN training to execute inference with both full-precision (32-bit floating-point) and low-precision (4-bit fixed-point) weights. To train SNNs for low-precision inference,

TABLE VI: Experimental results of the proposed LT-SNN on DVS-CIFAR10, CIFAR-100, and ImageNet-100 datasets. Except "VGG-9 (4-bit)", 32-bit weight precision is used for all results of prior works and this work.

Dataset	Method	SNN Architecture	# of Parameters	Weight Precision	Simulation Length	Top-1 Accuracy
DVS-CIFAR-10	ASF-BP [25]	VGG-Like	15.2M	32-bit	50	62.50%
	tdBN [9]	ResNet-19	12.31M	32-bit	10	67.80%
	ParamLIF [16]	VGG-Like	17.4M	32-bit	20	74.80%
	RecDis [10]	ResNet-19	12.31M	32-bit	10	72.42%
	TET [6]	VGG-Like	9.27M	32-bit	10	77.33%
	DSR [18]	VGG-11	9.34M	32-bit	30	75.70%
	Dspike [26]	ResNet-18	11.18M	32-bit	10	75.45%
	Spikeformer [27][2]	Spikformer-16-256	4.15M	32-bit	10	78.90%
	This work	VGG-11	**9.34M**	32-bit	**30**	**79.51%**
	This work	MobileNet-V1 (light)	**1.28M**	32-bit	**30**	**75.70%**
	This work	VGG-7	**1.91M**	32-bit	**30**	**80.20%**
	This work	VGG-9	**7.07M**	**4-bit**	**30**	**80.07%**
	This work	VGG-9	**7.07M**	32-bit	**10**	**79.10%**
	This work	VGG-9	**7.07M**	32-bit	**8**	**78.30%**
	This work	**Spikformer-16-256**	**4.15M**	32-bit	**10**	**79.00%**
CIFAR-10	Hybrid [28]	ResNet-20	12.3M	32-bit	10	92.54%
	tdBN [9]	ResNet-19	12.31M	32-bit	6	94.16%
	DSR [18][1]	ResNet-18	11.18M	32-bit	6	91.89%
	PSP [19]	VGG-19	12.20M	32-bit	400	93.97%
	Dspike [26]	ResNet-18	11.18M	32-bit	6	94.25%
	Spikeformer [27][2]	Spikformer-4-256	4.15M	32-bit	4	93.94%
	This work	**ResNet-19**	**12.31M**	32-bit	**2**	**94.19%**
	This work	**ResNet-19**	**12.31M**	32-bit	**6**	**94.56%**
	This work	**Spikformer-4-256**	**4.15M**	32-bit	**4**	**95.19%**
CIFAR-100	Hybrid [28]	VGG-16	11.11M	32-bit	5	69.67%
	DSR [18][1]	ResNet-18	11.18M	32-bit	6	68.33%
	PSP [19]	VGG-19	12.20M	32-bit	1100	73.58%
	Dspike [26]	ResNet-18	11.18M	32-bit	6	74.24%
	This work	ResNet-19	12.31M	32-bit	**6**	**74.82%**
	This work	ResNet-19	12.31M	32-bit	**2**	**72.78%**
ImageNet-100	TET [6][3]	ResNet-26	32.36M	32-bit	2	74.76%
	This work	ResNet-26	32.36M	32-bit	2	**75.58%**
ImageNet-1K	tdBN [9]	ResNet-34	32.36M	32-bit	6	63.72%
	TET [6][3]	ResNet-34	32.36M	32-bit	4	68.00%
	Dspike [26]	ResNet-34	32.36M	32-bit	6	68.19%
	This work	ResNet-34	32.36M	32-bit	4	**68.46%**

[1] The experiment is rigorously performed based on the open-sourced DSR [18] implementation with 6 time steps.
[2] The experiment is performed based on the open-sourced Spikeformer [27] implementation.
[3] The experiment is performed based on the open-sourced implementation of [6] with 100 epochs training and SGD optimizer.

TABLE VII: Experimental results of the proposed LT-SNN on N-Caltech101 and N-Cars datasets.

Dataset	Method	Representation	Accuracy	Pretrain
N-Caltech101	YOLE [34]	Histogram	70.02%	False
	EST [35]	Histogram	81.70%	True
	AsyNet [36]	Histogram	76.10%	False
	AEGNN [37]	Graph	66.80%	False
	This work	**Events**	**81.98%**	**False**
N-Cars	NDA [38]	Events	91.90%	False
	AsyNet [36]	Histogram	94.40%	False
	YOLE [34]	Histogram	92.70%	False
	AEGNN [37]	Graph	94.50%	False
	Object [39]	Binary Spikes	92.40%	False
	This work	**Events**	**95.02%**	**False**

we adopt the Statistics-Aware Weight Binning (SAWB) quantizer [40] to compress the layer-wise weights of the LT-SNN-VGG-9 network down to 4-bit precision.

c) Classification: We validate the proposed LT-SNN algorithm on both event-based classification datasets and conventional static datasets with RGB images. The LT-SNN-based ResNet-19 and spikeformer models are trained from scratch on CIFAR-10 and CIFAR-100 datasets. Extended from the ResNet-19 model, we apply the proposed LT-SNN algorithm to ResNet-26 and ResNet-34 and train these models on ImageNet-100 and ImageNet-1K. The proposed method is trained using PyTorch on four GPUs with distributed data parallelism. Regarding the event-based classification tasks, we apply the LT-SNN to both VGG-based and MobileNet-based encoders. The detailed model architectures are summarized in Table I. For both static and event-based datasets, we use the Adam optimizer with an initial learning rate of 0.001 reduced with cosine decay. We use a batch size of 128 for static datasets and 32 for event-based datasets. Table VI summarizes the performance of LT-SNN across all the mainstream benchmark datasets.

For the DVS-CIFAR10 dataset, compared to the current SoTA method [6], the 4-bit VGG-9 model trained by the proposed LT-SNN algorithm achieves 2.71% accuracy improvement with 1.31× fewer parameters and 10.48× model size reduction (MB). Furthermore, the proposed LT-SNN demonstrates consistently superior performance with different

TABLE VIII: Experimental results of the proposed LT-SNN on Prophesee Automotive Gen1 dataset.

Method	Model Architecture	SNN	Threshold	mAP
Asynet [36]	FB-Sparse	No	-	0.145
MatrixLSTM [41]	ResNet-19	No	-	0.3
RED [42]	RetinaNet	No	-	0.41
VGG-11+SSD [39]	**VGG+SSD-SNN**	Yes	Fixed	0.187
This work	**Custom-YoloV2-SNN**	**Yes**	**Fixed**	**0.122**
This work	**Custom-YoloV2-SNN**	**Yes**	**Learnable**	**0.298**

simulation lengths (from 8 time steps to 30 time steps).

For the CIFAR-10 dataset, LT-SNN also achieves new SoTA performance with ResNet-19. Similarly, for CIFAR-100, LT-SNN surpasses the existing SoTA accuracy using the identical training setup. With the ResNet-26 and ResNet-34 models, we also evaluate the performance of LT-SNN with the larger-scale ImageNet-100 and ImageNet-1k datasets. As shown in Table VI, LT-SNN outperforms the SoTA TET [6] and Dspike [26] by 0.82% and 0.27% respectively.

In addition to the CNN-based encoders, we also validate the LT-SNN algorithm with SpikeFormers [27] on DVS-CIFAR10 and CIFAR-10 datasets. Unlike how [27] applied augmentations to the DVS-CIFAR10 data, our work validates the performance of Spikeformer-based LT-SNN using the original input events without any augmentation. On top of the official code of Spikeformer, we train the same transformer architecture with LT-SNN and outperform spikeformer [27] by 0.10% and 1.25% from the reproduced baseline for DVS-CIFAR10 and CIFAR-10 datasets, respectively. Furthermore, we evaluate the proposed LT-SNN algorithm on N-CalTech101 [31] and N-Cars [30] datasets. As shown in Table VII, the proposed LT-SNN achieves new SoTA performance on both datasets with 5.88% (N-Caltech101, without pre-training) and 0.52% (N-Cars) accuracy improvements. Compared to the conventional histogram-based computation [34], [36] that use non-binary activations, LT-SNN enables end-to-end spike-based learning with superior performance and high hardware compatibility.

d) Impact of the Event Time Steps: SNNs require iterative computation and membrane potential accumulation, which motivated prior works to exploit the computation reduction with less number of time steps to represent the incoming event. We also validated LT-SNN with different simulated time steps on the DVS-CIFAR10 dataset. As shown in Table VI, the proposed LT-SNN achieves SoTA performance with reduced time steps and compact models, compared to the prior works. With the same 10 time steps as TET [6], LT-SNN achieves 1.77% accuracy improvement, without using any data augmentation. With the extended 30 time steps, LT-SNN achieves 80.20% SoTA accuracy on the DVS-CIFAR10 dataset.

e) Object Detection Task: In addition to image classification, we validate the proposed LT-SNN for object detection on a large-sized Prophesee Automotive Gen1 dataset [32]. With 228,123 bounding boxes for cars and 27,658 boxes for pedestrians, the Gen1 dataset [32] is considered as the most complex event-based computer vision task. Unlike prior works that used accumulated histogram [36], [39], [41] or graph-based input representation [37], we translate DVS events to pure binary frames and synchronize them with artificial actual ground truths from [42] for SNN training. The binary input events and intermediate spikes enable end-to-end binarized computing for LT-SNN, elevating the computation efficiency with a simplified data format. We use the custom LT-SNN-YoloV2 encoder (Table I) followed by Yolo loss to train the customized model. To avoid the gradient vanishing in deep SNN, we develop a shallow YoloV2 model by skipping one convolution block from the original architecture. In addition, we use stride 2 instead of 7 in the max-pooling layers to keep the salient features with the size of 223×287.

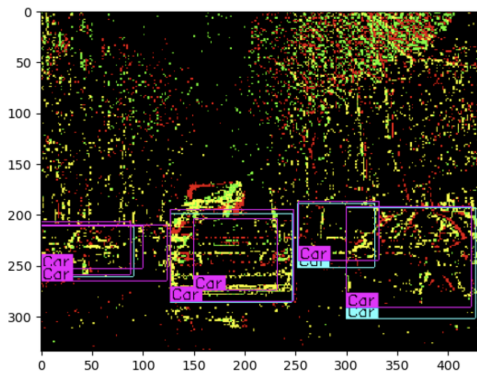

Fig. 6: Inference results of LT-SNN on Prophesee Automotive Gen1 dataset.

An example object detection result is illustrated in Figure 6, and Table VIII compares the LT-SNN based object detection results to the current SNN-based SoTA with a fixed potential threshold. Our custom LT-SNN-Yolov2 records SoTA mAP of 0.298 on the Prophesee Automotive Gen1 dataset.

VII. Conclusion

In this work, we present a novel SNN training algorithm with learnable threshold (LT-SNN), which optimizes the layer-wise threshold with direct SNN training. As one of the first studies on this topic, the proposed LT-SNN unleashes the firing constraints that were imposed in prior works. LT-SNN optimizes the performance of SNN without introducing any high-precision spike representations or learning constraints. The proposed method has been verified on a wide range of datasets, including both event-based and static image datasets for classification and object detection tasks. LT-SNN improves the state-of-the-art accuracy for DVS-CIFAR10 and N-Caltech dataset by 2.8% and 5.88% respectively together with 10.48× smaller model size. For object detection on the Prophesee Automotive Gen1 dataset, the LT-SNN outperforms SNN-based SoTA mAP by 0.11. Furthermore, the proposed method achieves the new SoTA results on CIFAR-10, CIFAR-100, ImageNet-100, and ImageNet-1k datasets, with both CNN-based encoder and spike-based vision transformer models.

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] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, A. Mascheroni, E. Reynaud, P. Mostafalu, F. Brady, L. Chotard, F. LeGoff, H. Takahashi, H. Wakabayashi, Y. Oike, and C. Posch, "A 1280×720 Back-Illuminated Stacked Temporal Contrast Event-Based Vision Sensor with 4.86μm Pixels, 1.066GEPS Readout, Programmable Event-Rate Controller and Compressive Data-Formatting Pipeline," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020.

[3] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, 2020.

[4] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, High-accuracy Spiking Deep Networks Through Weight and Threshold Balancing," in *International Joint Conference on Neural Networks (IJCNN)*, 2015.

[5] B. Han, G. Srinivasan, and K. Roy, "RMP-SNN: Residual Membrane Potential Neuron for Enabling Deeper High-accuracy and Low-latency Spiking Neural Network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] 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.

[7] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training Deep Spiking Neural Networks Using Backpropagation," *Frontiers in Neuroscience*, 2016.

[8] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct Training for Spiking Neural Networks: Faster, Larger, Better," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[9] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going Deeper with Directly-trained Larger Spiking Neural Networks," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[10] Y. Guo, X. Tong, Y. Chen, L. Zhang, X. Liu, Z. Ma, and X. Huang, "RecDis-SNN: Rectifying Membrane Potential Distribution for Directly Training Spiking Neural Networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[11] M. H. P. Kole and G. J. Stuart, "Is Action Potential Threshold Lowest in the Axon?" *Nature Neuroscience*, vol. 11, no. 11, 2008.

[12] R. Mozzachiodi and J. H. Byrne, "More than Synaptic Plasticity: Role of Nonsynaptic Plasticity in Learning and memory," *Trends in Neurosciences*, vol. 33, no. 1, 2010.

[13] P. Joshi and J. Triesch, "Rules for information maximization in spiking neurons using intrinsic plasticity," in *2009 international joint conference on neural networks*. IEEE, 2009.

[14] C. Huang, A. Resnik, T. Celikel, and B. Englitz, "Adaptive spike threshold enables robust and temporally precise neuronal encoding," *PLoS computational biology*, vol. 12, no. 6, 2016.

[15] D. Salaj, A. Subramoney, C. Kraisnikovic, G. Bellec, R. Legenstein, and W. Maass, "Spike frequency adaptation supports network computations on temporally dispersed information," *Elife*, vol. 10, 2021.

[16] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[17] A. Shaban, S. S. Bezugam, and M. Suri, "An adaptive threshold neuron for recurrent spiking neural networks with nanodevice hardware implementation," *Nature Communications*, vol. 12, no. 1, 2021.

[18] 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.

[19] Y. Chen, Y. Mai, R. Feng, and J. Xiao, "An adaptive threshold mechanism for accurate and efficient deep spiking convolutional neural networks," *Neurocomputing*, vol. 469, 2022.

[20] B. Rueckauer, I.-A. Lungu, Y. Hu, and M. Pfeiffer, "Theory and Tools for the Conversion of Analog to Spiking Convolutional Neural Networks," *arXiv preprint arXiv:1612.04052*, 2016.

[21] 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.

[22] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep Residual Learning in Spiking Neural Networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021.

[23] G. Shen, D. Zhao, and Y. Zeng, "Backpropagation with Biologically Plausible Spatiotemporal Adjustment for Training Deep Spiking Neural Networks," *Patterns*, 2022.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.

[25] H. Wu, Y. Zhang, W. Weng, Y. Zhang, Z. Xiong, Z.-J. Zha, X. Sun, and F. Wu, "Training Spiking Neural Networks with Acumulated spiking flow," in *Proceedings of the AAAI conference on Artificial Intelligence (AAAI)*, vol. 35, no. 12, 2021.

[26] 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*, vol. 34, 2021.

[27] Z. Zhou, Y. Zhu, C. He, Y. Wang, S. YAN, Y. Tian, and L. Yuan, "Spikformer: When spiking neural network meets transformer," in *The Eleventh International Conference on Learning Representations*, 2023.

[28] N. Rathi and K. Roy, "DIET-SNN: Direct Input Encoding with Leakage and Threshold Optimization in Deep Spiking Neural Networks," *arXiv preprint arXiv:2008.03658*, 2020.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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 32*. Curran Associates, Inc., 2019.

[34] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Asynchronous Convolutional Networks for Object Detection in Neuromorphic Cameras," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[35] D. Gehrig, A. Loquercio, K. G. Derpanis, and D. Scaramuzza, "End-to-end Learning of Representations for Asynchronous Event-based Data," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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

[37] S. Schaefer, D. Gehrig, and D. Scaramuzza, "AEGNN: Asynchronous event-based graph neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[38] Y. Li, Y. Kim, H. Park, T. Geller, and P. Panda, "Neuromorphic Data Augmentation for Training Spiking Neural Networks," *arXiv preprint arXiv:2203.06145*, 2022.

[39] L. e. a. Cordone, "Object detection with spiking neural networks on automotive event data," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022.

[40] J. Choi, S. Venkataramani, V. V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks," *Conference on Machine Learning and Systems (SysML)*, vol. 1, 2019.

[41] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "A Differentiable Recurrent Surface for Asynchronous Event-based Data," in *European Conference on Computer Vision (ECCV)*, 2020.

[42] 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)*, vol. 33, 2020.