Learning to Sparsify Differences of Synaptic Signal for Efficient Event Processing

Yusuke Sekikawa, Keisuke Uto
{ysekikawa,kuto}@d-itlab.co.jp
DENSO IT Laboratory
Shibuya, Shibuya-ku, Tokyo, Japan

Abstract

Neural network processors on edge devices need to process spatiotemporal data with low latency, which requires a large amount of multiply-accumulate operation (MAC). In this paper, we propose a difference-driven neural network framework for efficient video or event stream processing. Our framework achieves lower MAC by learning to sparsify the *temporal differences of synaptic signals* (TDSS) of proposed masked convolutional neural networks. By reducing the TDSS, MAC reduction is achieved in a unified manner by increasing the quantization step size, disconnecting synapses, and learning weights that respond sparsely to inputs. A novel quantizer is another key to realize unified optimization; the quantizer has a gradient called *macro-grad* that guides the step size to reduce the MAC by reducing the TDSS loss. Experiments conducted using a wide range of tasks and data (frames/events) show that the proposed framework can reduce MAC by a factor of 32 to 240 compared to dense convolution while maintaining comparable accuracy, which is several times better than the current state-of-the-art methods.

1 Introduction

Edge devices such as autonomous vehicles and mobile phones often use neural networks (NNs) to process spatiotemporal data such as videos. Processing such data independently in a frame-by-frame manner requires intensive computation.

Approaches for the efficient processing of spatiotemporal data from both algorithmic and processor perspectives are explored [32, 44, 46, 48]. Data from temporal proximity are highly correlated; that is, *differences* between each pixel from consecutive frames are highly sparse. The ΣΔ network [46] was proposed for efficient video processing; this is a reformulation of quantized neural networks into equivalent quantized recurrent neural networks. The network recursively updates its state and output using the sparse temporal difference. The sparse update mechanism results in lower multiply-accumulate operation (MAC). The model has been

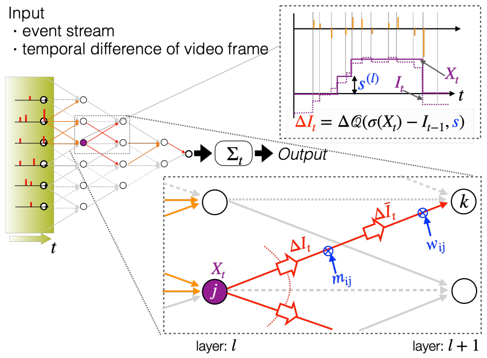

Figure 1: ***TDSS-aware training.*** In an mΣΔ network, only the masked difference $\Delta\bar{I}$ is processed. MAC could be reduced jointly by weight w, mask m, and quantization step size s. We realize the optimization by the novel quantizer and loss function.

proven effective for various tasks such as classification, object detection, and pose-estimation [44, 46, 48]. It has also been shown to be applicable for asynchronous event data [32, 69]. The ΣΔ networks provide very efficient inference, especially when mapped to processors that can take advantage of dynamic sparsity [7, 21, 23, 43]. These devices are becoming popular due to their efficiency and the advancement of CMOS technology.

We found that there is room for significant performance improvement of the network. In the existing method, the ΣΔ networks are first trained without quantization; then, the quantization step size is optimized for testing (*post-quantization*) either by greedy search ([32]) or by layer-wise stochastic gradient descent [46]. **1**) Increasing the step size reduces the MAC by making the differences sparser, but this is a tradeoff with accuracy. Joint optimization of weight and step size by end-to-end (E2E) learning seems promising; however, an existing technique for learning the step size of quantized neural networks such as LSQ [12] cannot train the step size of ΣΔ networks to reduce MAC. **2**) MAC could be further reduced by sparsifying synaptic connections; however, existing weight pruning techniques [25, 34, 67] is not applicable to learn the synaptic connection of ΣΔ networks to reduce MAC.

To this end, we propose a framework called *TDSS-aware training*, which significantly improves the MAC/accuracy tradeoff of ΣΔ networks with sparse connection by realizing joint optimization of weight, synaptic connection, and quantization step-size (Fig. 1). Our framework consists of two core inventions: **1**) A quantizer with a gradient *macro-grad* that guides the step size toward the direction of reducing MAC; **2**) A network module mΣΔ layer that could represent sparse connection and loss function using the *temporal difference of synaptic signal* (TDSS) which directly reflect MAC of mΣΔ networks.

We applied the proposed framework for different kinds of networks/tasks (including both frame-based and event-based data) and achieved accuracy comparable to the baseline using dense convolution while significantly reducing MAC up to 240x, which is 39x more efficient than state-of-the-art methods (SOTAs) utilizing temporal sparsity [32, 42, 46, 48, 69].

The joint optimization with TDSS lets the network select the computational path by activation, incurring fewer synaptic signals in subsequent layers. Therefore, a somewhat counterintuitive result for dense networks occurs; larger networks could learn to operate with lower MACs. We verified this by experiments.

2 Related Works

There has been a great deal of research on speeding up the inference of NNs, which includes network quantization [12, 30], low-rank approximating of weight [8, 38], designing efficient network structure [26, 62], and utilization of sparsity. In this section, we first review studies utilizing sparsity in weight/activation. Then we discuss studies focusing on sparsity in time for efficient spatiotemporal signal processing, which is the focus of our research.

Sparsity in weight/activation. A popular strategy for exploiting sparsity for efficient computation eliminates unnecessary operations involving zeros in weights and/or features. Weight pruning removes weights that are considered to have little impact on the output [10, 25, 34, 67]. NVidia's recent Ampere architecture supports structured sparsity in weight to accelerate convolution on GPUs. Sparsity in features is also utilized to accelerate convolution on CPUs [58]. Submanifold sparse convolutional (SSC) [20] is a new kind of sparse convolution that keeps the same sparsity after the convolution, which has recently been extended to process sparse event data asynchronously [42]. Efficient inference engine (EIE) [23] is a specifically designed ASIC to exploit the dynamic sparsity of the input feature maps to accelerate the inference of a neural network.

Sparsity in temporal difference. Inspired by a biological vision, event-based cameras [52] detect changes in luminance instead of sensing absolute brightness. This novel mechanism enables the efficient sparse sensing of spatiotemporal visual data [17]. Spiking NNs (SNNs) [19, 37, 53, 56, 65] are promising models for efficiently processing sparse data such as events or differences between video frames. However, SNNs have not been as successful as dense artificial NNs, mainly because of difficulties in training larger networks due to the non-differentiability [37, 56, 65] and the absence of large-scale hardware implementing SNN neurons such as leaky and fire (LIF), which involves more complex temporal dynamics.

The temporal sparsity also exists in frame-based video data by considering their temporal differences (e.g., adjacent frames). A common strategy to exploit the temporal sparsity of video is to propagate activation maps computed at key frames, thus avoiding computing expensive activation maps for each frame [27, 40, 45, 57, 68, 70]. Subsequent frames then re-use the spatially aligned activation maps via optical flow [68, 70], dynamic filters [40, 45], or self-attention [27].

The ΣΔ network was first proposed [46] for efficient video processing (Sec. 3.1). A concurrent work [44] extended it for gated recurrent units (GRU) [9] to model temporal dependencies. Later on, a simplified model (called the temporal difference (TD) network in [46]) and its extensions were applied for more complex tasks such as object detection [22, 48]. The same ΣΔ network was proposed in the SNN literature [32, 69] to process data from event-based cameras [52]. The ΣΔ networks enable low-power, low-latency inference, especially when mapped to devices that can take advantage of dynamic sparsity [7, 21, 23, 43], i.e., devices that can skip operations with zeros in weights and activation. Large-scale SoCs implementing the ΣΔ neuron is emerging on the market [43].

We believe the ΣΔ network is a promising candidate for processing sparse temporal signals because it is easy to train and has hardware-friendly simple temporal dynamics. Our work aims to extend this to achieve an optimal MAC/accuracy tradeoff by a novel mΣΔ network and realizing E2E training of the network by a novel quantizer and loss function.

3 Learning to Sparsify the Difference of a Synaptic Signal

3.1 Preliminary

In this preliminary section, we first formalize the ΣΔ network [46], which is the premise of this study, and then discuss the problems of existing approaches for reducing the network's MAC. Given a video sequence $[I_0^{(0)},\dots,I_t^{(0)},\dots]$[1], our goal is to get the corresponding output $[y_0,\dots,y_t,\dots]$, where y_t depends only on input at time t. The ΣΔ network is designed to efficiently process spatiotemporal signals by only processing changes of activation.

Quantized network. An l-th convolution (conv) layer of a quantized neural network is expressed as:

$$I_t^{(l+1)} = \sigma(\mathcal{Q}(w^{(l)} * I_t^{(l)} + b^{(l)}, s)), \tag{1}$$

where $w \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}\times k\times k}$ and $b \in \mathbb{R}^{C_{\text{out}}}$ are weights and biases for conv, σ is activation function, such as ReLU, $*$ represents a convolution, and $\mathcal{Q}$ is a quantizer for activation defined as:

$$\mathcal{Q}(x,s) = s\lfloor x/s \rceil, \tag{2}$$

where $d\lfloor\cdot\rceil$ is a round operation and s is quantization step size.

ΣΔ network. Eq. (1) is equivalent to the following quantized recurrent neural networks:

$$\Delta I_t^{(l+1)} = \mathcal{Q}(\sigma(X_t^{(l)}) - I_{t-1}^{(l+1)}, s), \tag{3}$$

where the internal state $X_t^{(l)}$ and $I_t^{(l+1)}$ is also updated recursively as follows,

$$X_t^{(l)} = X_{t-1}^{(l)} + w^{(l)} * \Delta I_t^{(l)} \tag{4}$$

$$I_t^{(l+1)} = I_{t-1}^{(l)} + \mathcal{Q}(\sigma(X_t^{(l)}) - I_{t-1}^{(l+1)}, s), \tag{5}$$

which is similar to sigma-delta modulation in digital signal processing. The I_t stores quantized activation map. In the recursion, X_0 and I_0 are initialized with zeros, and t starts from 1. When ΔI_t is sparse (i.e., most of its elements are zero), then $w * \Delta I_t$ in (4) is computed more efficiently than the dense operation of (1) when the network is mapped onto processors which could exploit dynamic sparsity (ignore zeros), such as EIE [23], IPU [21], TrueNorth [7], or neuron-flow [43], for inference.

Training the ΣΔ network. Eq. (3) suggests that computation could be reduced by increasing quantization step size s. In the conventional method, the network of (1) is first trained without quantization ($\mathcal{Q}$ is identity), w and b are fixed, and then s is optimized to increase the sparsity of ΔI. It was done by greedy search [32, 69] or by layer-wise stochastic gradient descent (SGD) [46] to reduce the *temporal difference of activation map* (TDAM):

$$L_{TDAM} = \sum_l |C_{\text{out}}^{(l+1)} k^{(l+1)} k^{(l+1)} \Delta I^{(l)}|_1 \tag{6}$$

3.2 TDSS-Aware Training

Problem statement. The quantized network described above is an approximation of trained non-quantized networks. Increasing the step size s reduces the MAC, but this is a tradeoff with accuracy. **1**) A joint optimization of weight w and step size s is expected to realize a much better MAC/accuracy tradeoff. However, as discussed in [46], the application of straight-through estimator (STE) [3] for s could not train the network in a fully E2E manner (Sec. 3.2.1). They, therefore, adopted layer-by-layer post-quantization. This is sub-optimal in the sense that s and w at the lower layer could not be optimized to reduce ΔI in higher layers. **2**) From another perspective, the existing ΣΔ network does not utilize the sparsity in w. Sparse weight could be represented by using binary masks [67]; however, reducing the TDAM of (6) could not directly reduce the MAC of ΣΔ networks with sparse weight.

Summary. To overcome these difficulties, we propose a framework called *TDSS-aware training*, which achieves significant MAC reduction over the existing *post-quantization* of ΣΔ networks. **1**) Regarding the quantizer, we contemplate the reasons for this and propose a quantizer with *macro-grad*, which realizes E2E joint optimization of weights, synaptic connections, and step size in a unified manner (Sec. 3.2.1). **2**) Regarding the training of synaptic connections, we propose the novel module called the mΣΔ layer to represent sparse connection and introduce a notion called TDSS where the MAC of the mΣΔ network could be directly reduced by minimizing it (Sec. 3.2.2).

[1]Note that the superscript in parentheses indicates the layer index. It is omitted to avoid clutter if it is obvious.

3.2.1 Gradient of Quantizer

The gradient of quantizer (2) w.r.t. x and s need to be computed for each quantized layer to realize E2E training using error backpropagation. The gradient w.r.t. x is easily computed using STE [3], which is 1 everywhere. Now, we focus on the gradient w.r.t. s. Consider two points x_t, x_{t-1} and $\mathcal{L} := |\hat{x}_t - \hat{x}_{t-1}|$ (**a**, **b** in Fig. 2). The gradient of $\mathcal{L}$ w.r.t. s is as follows:

$$\frac{\partial \mathcal{L}}{\partial s} = \operatorname{sgn}(\hat{x}_t - \hat{x}_{t-1}) \left(\frac{\partial \hat{x}_t}{\partial s} - \frac{\partial \hat{x}_{t-1}}{\partial s} \right) \quad (7)$$

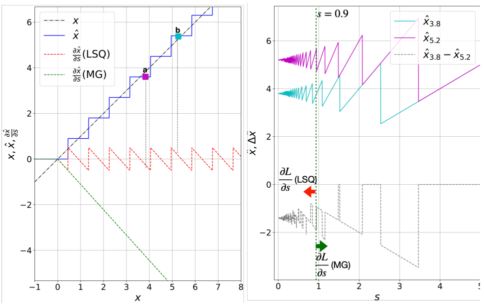

Figure 2: **Gradient of quantizer.** The gradient of the proposed *macro-grad* (MG) is compared with the gradient of LSQ [12] when $s = 0.9$ (Left). Quantized value $\hat{x}$ for different step size s (Right). $x = 3.8$, $x = 5.2$ corresponds to **a**, **b** on the left.

where $\hat{x}_t = \sigma(Q(x_t, s))$. Note that the $\mathcal{L}$ corresponds to MAC of $\Sigma\Delta$ networks, and recall that our goal is to reduce the MAC. In the following, for simplicity, let the $\sigma :=$ ReLU, and assume $x_t > x_{t-1} > 0$.

LSQ. Considering the gradient of (2) w.r.t. s in each quantization bin, we get the following:

$$\frac{\partial \hat{x}}{\partial s} = -x/s + \lfloor x/s \rceil \quad (8)$$

It is the same as the LSQ gradient [12] (without value range Q_n and Q_p), which is widely used for learning step size of quantized neural networks. Since this gradient is correct locally, the LSQ can indeed decrease $\mathcal{L}$, albeit very slightly. In the example of Fig. 2, the gradient at **a**, **b** are ≈ -0.22, $\approx +0.22$ respectively, and $\partial\mathcal{L}/\partial s \approx +0.44$. Therefore, in this case, $\mathcal{L}$ is reduced by decreasing s (making the quantization finer). The gradient guide s to reduce $\mathcal{L}$, but it does not always coincide with the direction to make $\mathcal{L}$ to be zero (equivalent to sparsify ΔI), as shown in Fig. 2 (right). If x is random value from uniform distribution in $[0, +\infty]$, then the gradient of (8) distribute uniformly in $[-0.5, +0.5]$; therefore, $\mathbb{E}(\partial\mathcal{L}/\partial s) = 0$. It means that LSQ is not capable of guiding s to reduce $\mathcal{L}$ ($\sim$MAC) substantially; this is also experimentally verified in the ablation study (Sec. 4.2).

Regularizer of s. It is possible to increase s by incorporating additional loss term (regularizer) such as $|1/s|$. In the case of LSQ, the gradient of the quantization error $|\hat{x} - x|$ w.r.t. s is $|-x/s + \lfloor x/s \rceil|$. A smaller task loss is generally realized by smaller quantization error, so s tends to become smaller by minimizing the task loss. Therefore, they will balance somewhere; however, this does not take each neuron's MAC (TDSS or TDAM loss) into account, resulting in a sub-optimal MAC/accuracy tradeoff. Dynamically adjusting the weight for the regularizer using the MAC of each neuron is very difficult to realize. Furthermore, the hyper-parameter for the regularizer makes the optimization and fair evaluation difficult.

Macro-grad. If the two points are not in the same quantization bin, then $\mathcal{L} > 0$. By gradually increasing s (while fixing x_t and x_{t-1}), $\hat{x}_{t-1}$ and $\hat{x}_t$ will fall into same bin, and the $\mathcal{L}$ becomes 0 (Fig. 2, right). That is, in a macroscopic point of view, $\partial\mathcal{L}/\partial s$ must be negative. Based on this observation, we propose a quantizer with a gradient called *macro-grad*,

$$\frac{\partial \hat{x}}{\partial s} = -\frac{\xi(x)}{s}, \quad (9)$$

where ξ could be any monotonically increasing function. We use the identity function for simplicity. In a microscopic view, it does not yield precise gradients as LSQ of (8), and it sometimes increases $\mathcal{L}$; however, in a macroscopic point of view, the gradient induces x_t and x_{t-1} to fall into the same quantization bin by increasing s. The gradient $\partial\mathcal{L}/\partial s$ is always negative for non-zero $\mathcal{L}$ since ξ is a monotonic function. In the example of Fig. 2, the gradient at **a**, **b** are ≈ -4.2, ≈ -5.8 respectively, and $\partial\mathcal{L}/\partial s \approx -1.6$. The gradient makes s larger; it may not reduce $\mathcal{L}$ immediately; however, eventually $\mathcal{L}$ become zero as they fall into the same bin. Now, consider a difference between quantized value $\hat{x}$ and target value x_{tgt}. The gradient of $|\hat{x} - x_{\text{tgt}}|$ w.r.t. s is positive if $\hat{x} < x_{\text{tgt}}$ and negative otherwise. The gradient reduces the difference in both cases. Therefore, the *macro-grad* automatically balances between accuracy (task loss) and MAC (TDAM or TDSS loss). The above discussion is also valid for the more general TDSS loss of (12) for the m$\Sigma\Delta$ (Sec. 3.2.2) network, and the combination of TDSS and *macro-grad* significantly reduces MAC of the network by jointly optimizing weight, mask, and step size. We experimentally verified that s becomes larger by reducing (6) or (12) in the case of the *macro-grad* while LSQ does not (Sec. 4.2).

3.2.2 Masked ΣΔ Network and TDSS

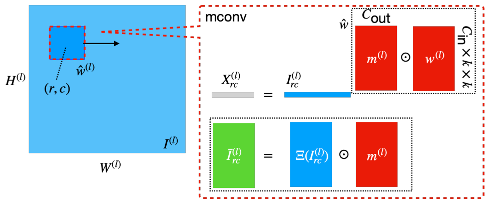

Figure 3: **Synaptic signal.** Synaptic signal $\bar{I}$ is Hadamard product of unfolded input $\Xi(I)$ and mask m. Sparsifying TDSS $\Delta\bar{I}$ directly reduces MAC.

In the $\Sigma\Delta$ network, MAC is proportional to the total number of signals triggered when the input differential signal (events or difference of consecutive video frame) is presented to the network. In each neuron, an update signal (quantized difference of activation map) is triggered when the difference of activation map exceeds the step size s. The update signal is sent to all the connected neurons in the subsequent layer (Fig. 1). Therefore, even if some neuron fires, induced MAC would be small if the neuron's connection is sparse. In order to reduce MAC by sparsifying synaptic connections, we propose the m$\Sigma\Delta$ network that incorporates a learnable mask. The m$\Sigma\Delta$ updates its internal states as follows:

$$X_t^{(l)} = X_{t-1}^{(l)} + \hat{w}^{(l)} * \Delta I_t^{(l)} = X_{t-1}^{(l)} + (m^{(l)} \odot w^{(l)}) * \Delta I_t^{(l)}, \tag{10}$$

where, mask $m \in \{0,1\}^{C_{\text{out}} \times C_{\text{in}} kk}$ represents the binary synaptic connection between neurons in the subsequent layer. After training, synaptic weights having non-zero masks are mapped to the hardware. The sparse weight reduces both MAC and memory footprints. However, the TDAM loss of (6) does not yield a precise gradient for each element of the mask w.r.t. MAC. It enhances overall sparsity in w, but it could not take the relationship with ΔI into account, which results in a sub-optimal mask. To this end, we introduce a novel quantity called the *synaptic signal*, $\bar{I}_t$ defined as follows:

$$\bar{I}_t^{(l)} := \Xi(I_t^{(l)}) \odot m^{(l+1)} \tag{11}$$

$$\Delta\bar{I}_t^{(l)} := \bar{I}_t^{(l)} - \bar{I}_{t-1}^{(l)}, \tag{12}$$

where $\odot$ is Hadamard product, and Ξ is unfolding[2] function (Fig. 3). The TDSS is defined as a temporal difference of two *synaptic signals* of temporal proximity, as shown in (12). The

[2]Flatten each sliding local k^2 block of input and copies values along the output channel dimension($I_t \in \mathbb{R}^{C_{\text{in}} \times WH} \rightarrow \Xi(I)_t \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} kk \times WH}$. The Hadamard product is applied along each spatial location (HW dimension).

sparsity of TDSS directly corresponds to the MAC of the m$\Sigma\Delta$ network. In fact, the MAC of the m$\Sigma\Delta$ network is equal to the l_0 norm of TDSS. We define TDSS loss $\mathcal{L}_{TDSS}$ as the l_1 norm of TDSS, which is differentiable. Our m$\Sigma\Delta$ networks are trained with a combination of $\mathcal{L}_{TDSS}$ and task loss $\mathcal{L}_{tgt}$, such that $\mathcal{L} = \mathcal{L}_{tgt} + \eta\mathcal{L}_{TDSS}$, where η is a hyper-parameter to balance the sparsity of TDSS with accuracy. By learning to reduce $\mathcal{L}_{TDSS}$, the MAC of the m$\Sigma\Delta$ network could be reduced.

Parameterization of weight and mask. We parameterize the mask m and weight w using the same parameter w to reduce memory footprints and improve training stability. The masked kernel $\hat{w}$ for (10) and mask m to evaluate the TDSS loss of (12) is computed as:

$$\hat{w}^{(l)} = \text{softshrink}(w^{(l)}, \gamma) \tag{13}$$

$$m^{(l)} = \text{softshrink}(w^{(l+1)}, \gamma) \neq 0 \tag{14}$$

This formulation using softshrink enables us to represent an arbitrarily small masked weight $\hat{w}$, which is not the case for simple weight masking by magnitude [10, 25, 67]. The weight w receives gradient signals from the two sources. Both gradients are specifically designed such that neurons are masked to reduce the TDSS loss, or they could become active again to achieve better accuracies to reduce task loss. The threshold γ is a fixed parameter that is the same for both functions, and we set $\gamma = 0.1\sqrt{6/C_{\text{out}}}$. Refer to Supp. F for more detail about the parameterization, the gradients, and the initialization scheme.

Memory efficient evaluation of TDSS. A large amount of memory ($\mathbb{R}^{C_{\text{out}}C_{\text{in}}kkWH}$ for a single layer) is required to explicitly compute TDSS of (12). By noticing,

$$\sum_l ||\Delta\bar{I}^{(l)}||_1 = \sum_l || \, |\Xi(\Delta I^{(l)})| \odot m^{(l+1)}||_1 = \sum_l || \, |\Delta I^{(l)}| * m^{(l+1)}||_1, \tag{15}$$

we can directly accumulate the sum into a scalar value. This trick obviates the need to allocate large amounts of memory.

Random sampling of TDSS. Although the technique of (15) significantly reduces the memory footprint, computing TDSS requires the additional computation regarding mask m and consumes memory for backpropagation. To alleviate this problem, we randomly sample a few layers per mini-batch to evaluate the TDSS. The TDSS of the selected layer is computed online, avoiding the explicit computation in the other layers.

3.3 Extension to Event Data

Event data. Event camera triggers an event when the (logarithmic) luminance change exceeds a predetermined threshold $s^{(0)}$ at each pixel. As input representation $I_t^{(0)}$, we adopt histogram of fixed number of events τ. In this representation, the value on each pixel is the event count for each polarity, which is visualized in Fig. 4 (top).

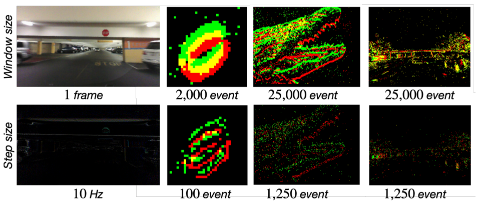

Figure 4: **Input representation.** Dense input $I_t^{(0)}$ for training (Top). Sparse input $\Delta I_t^{(0)}$ for testing (Bottom).

Asynchronous update. Given a single event, the difference in input $\Delta I_{t,p}^{(0)}$ for polarity $p \in \{-1, +1\}$ is computed as follows:

$$\Delta I_{t,p}^{(0)} = I_{t,p}^{(0)} - I_{t-\delta t,p}^{(0)}, \tag{16}$$

where δt is the time elapsed since the last event has been observed. This difference is very sparse; only two pixels are nonzero on average (some pixels may go out from the accumulation window τ). The update rule of the m$\Sigma\Delta$ network in Sec. 3.2 could directly applied when we interpret $(t-1)$ as $(t-\delta t)$. The output and the state of the m$\Sigma\Delta$ network are updated asynchronously by feeding the difference triggered by a single event. This update, using very sparse $\Delta I^{(0)}$, is much more efficient than processing dense $I^{(0)}$ using (1).

Batch update. It is also possible to update the network using multiple events at once (processing step size ν). In this scenario, δt is the time elapsed since the oldest event in the batch has been observed. An important fact is, MAC is not proportional to the number of events ν processed at once. Given ν events, the MAC for processing them one by one is higher than processing them at once. It is partly because $\Sigma\Delta$ or m$\Sigma\Delta$ network could encode multiple bits in ΔI rather than 1-bit of pure SNN. Therefore, the batch update is more efficient than the asynchronous update when output at the input event rate is unnecessary. The asynchronous update scenario is the particular case where $\nu = 1$. Note that output at the rate of events (it could be more than 10^7-10^8 Hz) is usually too much and unnecessary, even for latency-critical apps [50].

MAC evaluation. We evaluate MAC by changing ν, including the special case of asynchronous update where $\nu = 1$. Obviously, the computational gain over dense operation of (1) becomes prominent when small ν is used. Nevertheless, we consider the batch update scenario is particularly important for practical application; we will mainly discuss MAC when $\nu \gg 1$.

Processing step size during training. The network is learned to reduce MAC by using TDSS loss. We choose processing step size for the TDSS loss $\nu_{\text{train}} \gg 1$, considering the practical scenario. The ν_{train} are visualized in Fig. 4 (bottom) for each dataset.

Other input representation. Other options exist for event representation [1, 35], which could be updated asynchronously for each incoming event. Moreover, it could be learned E2E [5, 18, 64] using task loss. This paper uses histograms for simplicity, and the exploration of learning input representations is left for future work.

4 Experiments

Our experiments are divided into two parts. First, we benchmark our framework on different kinds of tasks and data types: steering angle prediction (video), handwriting recognition (event), object recognition (event), and object detection (event). We achieved a significant reduction in MAC compared to dense networks while maintaining comparable accuracy; this is several times better than existing SOTAs (Sec. 4.1). Next, we present an extensive ablation study to reveal important elements for achieving the very low MAC (Sec. 4.2).

Table 1: **Experimental setup.** bs is batch size for training, epoch is training epochs, lr is initial learning rate, η_{step}/η_{thr} are TDSS scheduling parameter, size is spatial size of input representation, τ is number events for input representation, and ν_{train} is processing step size for TDSS loss during training.

	PilotNet	N-MNIST	N-Caltech	Gen1 Autom
bs	64	64	64	64
epoch	1,000	1,000	10,000	1,000
lr	1E-3	1E-3	1E-3	1E-3
η_{step}	4.0	2.0	0.1	1.0
η_{thr}	0.02	0.02	0.29	0.872
size	66×200	34×34	191×255	223×287
τ	-	2,000	25,000	25,000
ν_{train}	10	100	1,250	1,250

Table 2: **Experimental result.** We compare ours (red) with *Dense* (gray), *Asyc-SSC* [42] (green), and $\Sigma\Delta$-*TDAM* [46] (cyan). In quantizer ($\mathcal{Q}$), LW represents the layer-wise *post-quantization* of [46], MG represents quantization with the proposed *macro grad*, and LSQ is from [12] without a value range. The processing step size (frame rate or a number of events) for TDSS during training is shown in **bold**.

				PilotNet (frame)					**N-MNIST** (event)				
Processing step size $\nu \Rightarrow$					480	120	**10**			1	10	**100**	
Average processing rate (Hz) $\Rightarrow$					480	120	10			2.5E+04	2.5E+03	2.5E+02	
neuron	kernel	loss	$\mathcal{Q}$	MSE↓	MAC ($\times 10^{-6}$)↓			gain	Acc.↑	MAC ($\times 10^{-6}$)↓			gain
Dense		-	-	1.441	2.8E+01	2.8E+01	2.8E+01	1.0	0.991	1.7E+01	1.7E+01	1.7E+01	1.0
SSC	-	-	-	-	-	-	-	-	0.988	2.3E+00	2.0E+00	5.6E+00	3.1
$\Delta\Sigma$	conv	TDAM	LW	1.713	3.5E-02	1.8E+00	4.6E+00	6.1	0.989	8.3E-02	8.3E-02	1.8E+00	9.7
$\Delta\Sigma$	mconv	TDSS	MG	1.427	7.2E-02	4.4E-02	1.4E-01	195.1	0.989	9.5E-03	5.6E-02	2.1E-01	81.5
$\Delta\Sigma$	mconv (x2)	TDSS	MG	2.086	7.0E-02	4.3E-02	1.4E-01	196.6	0.990	8.7E-03	5.4E-02	2.1E-01	82.4
$\Delta\Sigma$	mconv (x4)	TDSS	MG	1.977	6.1E-02	3.7E-02	1.3E-01	217.0	0.991	6.9E-03	5.0E-02	1.8E-01	98.3
$\Delta\Sigma$	mconv (x8)	TDSS	MG	1.969	6.1E-02	3.2E-02	1.2E-01	238.3	0.991	6.4E-03	4.9E-02	1.7E-01	99.5
$\Delta\Sigma$	mconv	TDSS	LSQ	2.103	8.7E-02	1.2E-01	1.9E-01	145.9	0.991	3.0E-02	1.4E-01	3.2E-01	54.5
$\Delta\Sigma$	mconv	TDAM	MG	2.722	8.1E-01	1.8E+00	4.5E+00	6.3	0.989	1.9E-02	1.1E-01	4.8E-01	36.0
$\Delta\Sigma$	conv	TDSS	MG	1.725	6.7E-01	1.6E+00	4.1E+00	6.8	0.990	1.5E-02	8.6E-02	3.6E-01	48.1

				N-Caltech101 (event)					**Gen1 Automotive** (event)				
Processing step size $\nu \Rightarrow$					1	10	**1250**			1	10	**1250**	
Average processing rate (Hz) $\Rightarrow$					6.0E+05	6.0E+04	4.8E+03			9.7E+06	9.7E+05	7.7E+03	
neuron	mask	loss	$\mathcal{Q}$	Acc.↑	MAC ($\times 10^{-6}$)↓			gain	mAP↑	MAC ($\times 10^{-6}$)↓			gain
Dense		-	-	0.733	8.1E+02	8.1E+02	8.1E+02	1.0	0.128	1.2E+03	1.2E+03	1.2E+03	1.0
SSC	-	-	-	0.712	1.0E+02	2.3E+02	5.4E+02	1.5	0.119	1.0E+02	2.2E+02	4.9E+02	2.4
$\Delta\Sigma$	conv	TDAM	LW	0.731	5.4E+01	8.9E+01	1.8E+02	4.4	0.124	1.2E+02	2.0E+02	2.8E+02	4.3
$\Delta\Sigma$	mconv	TDSS	MG	0.727	6.0E+00	1.0E+01	2.0E+01	40.6	0.124	1.4E+01	2.6E+01	4.0E+01	30.3
$\Delta\Sigma$	mconv (x2)	TDSS	MG	0.726	5.5E+00	9.4E+00	1.9E+01	43.7	0.126	1.3E+01	2.4E+01	3.7E+01	32.8
$\Delta\Sigma$	mconv	TDSS	LSQ	0.724	1.0E+01	1.9E+01	2.7E+01	29.8	0.125	3.1E+01	6.1E-01	1.2E+00	14.1

4.1 Comparison with SOTA

Experimental setup. For all the experiments, we compare ours (m$\Sigma\Delta$-*TDSS*) with a network using dense convolution (*Dense*) and SOTA model ($\Sigma\Delta$-*TDAM*) of [46]. We also compare ours with another competitive method for event data based on Asynchronous-SSC [42] (*Asyc-SSC*) for experiments using event data. All the models share the same network configuration and settings unless otherwise stated. We use AdamW optimizer [41] for all experiments. For the frame-based dataset, we adopted the frame-to-event conversion method of [69]. For the event-based dataset, we adopted the two-channel histogram representation used in [42]. The $\Sigma\Delta$-*TDAM* [46] used learned weights from *Dense*. The step size of the quantizer is optimized per layer using the TDAM of (6). Our m$\Sigma\Delta$-*TDSS* used the mconv layer, which could represent the sparse synaptic connection. It is trained with the TDSS of (12) from scratch using the quantizer with the *macro-grad* of (9). For our model, we also evaluated the network having κ times larger input/output channels for mconv layers (except FC layer); thus, MAC using the dense computation of (1) is approximately κ^2 times larger than the original network ($\kappa = 1$). Both $\Sigma\Delta$-*TDAM* and m$\Sigma\Delta$-*TDSS* are trained with a specific processing step size during training (ν_{train}) and evaluated on three different step sizes ν. The TDSS/TDAM weights η are scheduled adaptively using validation result, increased by η_{step} when the validation error falls below a predetermined threshold η_{thr}. The quantization step sizes s are initialized with 2^{-10} for all layers. The experimental setup is summarized in Tab.1. Refer to Supp. H-I for Pytorch [49] implementation of propose *macro-grad* and TDSS loss.

Results. The results are summarized in Tab.2. We trained and evaluated all experiments three times using the different initialization of weight and reporting its mean. Our framework achieved an accuracy level similar to that of the *Dense* while achieving a significant reduction of MAC: 240x for Nvidia PilotNet [51], 99x for N-MNIST [47], 43x for N-Caltech101

[47], and 33x for Gen1 Automotive [63], which is 8 to 39 times more efficient than the SOTA model of [46] ($\Sigma\Delta$-*TDAM*) designed to process spatiotemporal data efficiency. Moreover, our model realizes lower MAC on different processing step sizes used during training. Refer to Supp. A for a more detailed setup, including a description of the dataset, network architecture, and additional comparison with more diverse methods.

4.2 Ablation study

Table 2 (under the dashed line) summarizes the ablation results for the factors that enable a significant MAC reduction of the proposed *TDSS-aware training*, specifically, TDSS loss, mask, *macro-grad*, and network size. **1**) By changing loss from TDAM to TDSS, the MAC/accuracy tradeoff improves up to 31x. **2**) By changing conv to mconv the MAC/accuracy tradeoff improves up to 29x. In this case, i.e., when all conv weights are connected, TDSS of (12) and TDAM of (6) are equivalent. **3**) By changing the gradient of quantizer from LSQ [12] to *macro-grad*, the MAC/accuracy tradeoff improves up to 2.1x. During training, s becomes larger as TDSS decreases when the *macro-grad* is used, while it remains almost unchanged when LSQ is used. **4**) Comparing our model with different sizes shows interesting results; larger models achieve lower MAC. This supports our assumption that by learning weight, mask, and step size to reduce the TDSS, the network is learned to select the computational path by activation, which incurs sparse difference in synaptic signals in the following layers. Refer to Supp. B-C for additional results on the ablation study.

5 Conclusion

This paper presents a framework called *TDSS-aware training* that realized E2E training of proposed m$\Sigma\Delta$ network by the quantizer equipped with *macro-grad* and loss function called TDSS. Our framework reduces MAC by more than two orders of magnitude than a dense network without sacrificing accuracy.

5.1 Limitations and Future Works

In the following, we describe some limitations of the proposed framework and also discuss possible future works.

Challenges for real hardware. Our current model assumes arbitrary step size and weight has 32-bit floating-point precision. We may achieve further efficiency by taking into account the specification of actual H/W for inference. For example, quantization with a power of two or convolution with 4- or 8-bit fixed-point precision would be more hardware friendly [30].

Combination with Gumbel gate. Based on a recurrent model similar to $\Sigma\Delta$ neuron[3], Skip-Conv [22] utilizes a mechanism called the *Gumbel gate* to adaptively estimate pixel-level gates to skip computation on pixels that are not relevant to the output. We expect adopting their gate to our framework could further improve the efficiency.

[3]Their model is based on TD neuron [46, 48], which is a simplification of $\Sigma\Delta$. Refer to the supplement (Sec. E) for detail about the TD model.

References

[1] R Baldwin, Ruixu Liu, Mohammed Almatrafi, Vijayan Asari, and Keigo Hirakawa. Time-ordered recent event (tore) volumes for event cameras. *arXiv preprint arXiv:2103.06108*, 2021.

[2] Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in neural information processing systems*, pages 9368–9378, 2018.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL `http://dblp.uni-trier.de/db/journals/corr/corr1308.html#BengioLC13`.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[5] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. Matrix-lstm: a differentiable recurrent surface for asynchronous event-based data.

[6] Marco Cannici, Marco Ciccone, Andrea Romanoni, and M. Matteucci. Asynchronous convolutional networks for object detection in neuromorphic cameras. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1656–1665, 2019.

[7] Andrew S. Cassidy, Paul Merolla, John V. Arthur, Steve K. Esser, Bryan Jackson, Rodrigo Alvarez-Icaza, Pallab Datta, Jun Sawada, Theodore M. Wong, Vitaly Feldman, Arnon Amir, Daniel Ben-Dayan Rubin, Filipp Akopyan, Emmett McQuinn, William P. Risk, and Dharmendra S. Modha. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10, 2013. doi: 10.1109/IJCNN.2013.6707077.

[8] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[9] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[10] Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4860–4874, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[11] Gamaleldin Elsayed, Prajit Ramachandran, Jonathon Shlens, and Simon Kornblith. Revisiting spatial invariance with low-rank local connectivity. In Hal Daumé III and Aarti

Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2868–2879, Virtual, 13–18 Jul 2020. PMLR. URL http://proceedings.mlr.press/v119/elsayed20a.html.

[12] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2019.

[13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[14] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[15] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975.

[16] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[17] G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. doi: 10.1109/TPAMI.2020.3008413.

[18] D. Gehrig, A. Loquercio, K. Derpanis, and D. Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5632–5642, 2019. doi: 10.1109/ICCV.2019.00573.

[19] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[20] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018.

[21] GraphCore. Graphcore. https://www.graphcore.ai/.

[22] Amirhossein Habibian, Davide Abati, Taco S. Cohen, and Babak Ehteshami Bejnordi. Skip-convolutions for efficient video processing. *CoRR*, abs/2104.11487, 2021. URL https://arxiv.org/abs/2104.11487.

[23] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[25] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[26] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. URL http://arxiv.org/abs/1704.04861. cite arxiv:1704.04861.

[27] Ping Hu, Fabian Caba, Oliver Wang, Zhe Lin, Stan Sclaroff, and Federico Perazzi. Temporally distributed networks for fast video semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8818–8827, 2020.

[28] David H Hubel and TN Wiesel. Shape and arrangement of columns in cat's striate cortex. *The Journal of physiology*, 165(3):559, 1963.

[29] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[30] Sambhav Jain, Albert Gural, Michael Wu, and Chris Dick. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 112–128, 2020. URL https://proceedings.mlsys.org/paper/2020/file/e2c420d928d4bf8ce0ff2ec19b371514-Paper.pdf.

[31] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

[32] Mina A Khoei, Amirreza Yousefzadeh, Arash Pourtaherian, Orlando Moreira, and Jonathan Tapson. Sparnet: Sparse asynchronous neural network execution for energy efficient inference. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 256–260. IEEE, 2020.

[33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[34] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5544–5555. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/kusupati20a.html.

[35] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman. Hots: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on Pattern*

Analysis and Machine Intelligence, 39(7):1346–1359, 2017. doi: 10.1109/TPAMI.2016.2574707.

[36] Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19:143–155, 1989.

[37] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10:508, 2016. ISSN 1662-453X. doi: 10.3389/fnins.2016.00508. URL https://www.frontiersin.org/article/10.3389/fnins.2016.00508.

[38] Chong Li and CJ Richard Shi. Constrained optimization based low-rank approximation of deep neural networks. In *European Conference on Computer Vision*, pages 746–761. Springer, 2018.

[39] Fei-Fei Li, Robert Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Comput. Vis. Image Underst.*, 106(1):59–70, 2007. URL http://dblp.uni-trier.de/db/journals/cviu/cviu106.html#Fei-FeiFP07.

[40] Yule Li, Jianping Shi, and Dahua Lin. Low-latency video semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5997–6005, 2018.

[41] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[42] Nico Messikommer, Daniel Gehrig, Antonio Loquercio, and Davide Scaramuzza. Event-based asynchronous sparse convolutional networks. 2020. URL http://rpg.ifi.uzh.ch/docs/ECCV20_Messikommer.pdf.

[43] O Moreira, Amirreza Yousefzadeh, F Chersi, A Kapoor, R-J Zwartenkot, P Qiao, G Cinserin, Mina A Khoei, M Lindwer, and Jonathan Tapson. Neuronflow: A hybrid neuromorphic–dataflow processor architecture for ai workloads. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 01–05. IEEE.

[44] Daniel Neil, Jun Haeng Lee, Tobi Delbruck, and Shih-Chii Liu. Delta networks for optimized recurrent network computation. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2584–2593. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/neil17a.html.

[45] Xuecheng Nie, Yuncheng Li, Linjie Luo, Ning Zhang, and Jiashi Feng. Dynamic kernel distillation for efficient pose estimation in videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6942–6950, 2019.

[46] Peter O'Connor and Max Welling. Sigma delta quantized networks. *arXiv preprint arXiv:1611.02024*, 2016.

[47] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9:437, 2015. ISSN 1662-453X. doi: 10.3389/fnins.2015.00437. URL `https://www.frontiersin.org/article/10.3389/fnins.2015.00437`.

[48] Bowen Pan, Wuwei Lin, Xiaolin Fang, Chaoqin Huang, Bolei Zhou, and Cewu Lu. Recurrent residual module for fast inference in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1536–1545, 2018.

[49] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learn` `pdf`.

[50] Etienne Perot, Pierre de Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi. Learning to detect objects with a 1 megapixel event camera. *arXiv preprint arXiv:2009.13436*, 2020.

[51] PilotNet. Pilotnet dataset. `https://github.com/lhzlhz/PilotNet`, 2018.

[52] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. High-dr frame-free pwm imaging with asynchronous aer intensity encoding and focal-plane temporal redundancy suppression. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 2430–2433. IEEE, 2010.

[53] Bipin Rajendran, Abu Sebastian, Michael Schmuker, Narayan Srinivasa, and Evangelos Eleftheriou. Low-power neuromorphic hardware for signal processing applications. *IEEE SIGNAL PROCESSING MAGAZINE*, 1053(5888/18), 2018.

[54] Bharath Ramesh, Hong Yang, Garrick Orchard, Ngoc Anh Le Thi, Shihao Zhang, and Cheng Xiang. Dart: distribution aware retinal transform for event-based cameras. *IEEE transactions on pattern analysis and machine intelligence*, 42(11):2767–2780, 2019.

[55] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[56] Oran Shayer, Dan Levi, and Ethan Fetaya. Learning discrete weights using the local reparameterization trick. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BySRH6CpW`.

[57] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In *European Conference on Computer Vision*, pages 852–868. Springer, 2016.

[58] Shaohuai Shi and Xiaowen Chu. Speeding up convolutional neural networks by exploiting the sparsity of rectifier units. *arXiv preprint arXiv:1704.07724*, 2017.

[59] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 1412–1421. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/82f2b308c3b01637c607ce05f52a2fed-Paper.pdf.

[60] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[61] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1731–1740, 2018.

[62] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[63] Pierre Tournemire, Davide Nitti, Etienne Perot, Davide Migliore, and Amos Sironi. A large scale event-based detection dataset for automotive, 2020.

[64] Stepan Tulyakov, Francois Fleuret, Martin Kiefel, Peter Gehler, and Michael Hirsch. Learning an event sequence embedding for dense event-based deep stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[65] Aaron R Voelker, Daniel Rasmussen, and Chris Eliasmith. A spike in performance: Training hybrid-spiking neural networks with quantized activation functions. *arXiv preprint arXiv:2002.03553*, 2020.

[66] Chr Von der Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14(2):85–100, 1973.

[67] Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. *Advances in Neural Information Processing Systems*, 32:2684–2694, 2019.

[68] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1278–1287. IEEE, 2019.

[69] Amirreza Yousefzadeh, Mina A Khoei, Sahar Hosseini, Priscila Holanda, Sam Leroux, Orlando Moreira, Jonathan Tapson, Bart Dhoedt, Pieter Simoens, Teresa Serrano-Gotarredona, et al. Asynchronous spiking neurons, the natural key to exploit temporal sparsity. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(4): 668–678, 2019.

[70] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2349–2358, 2017.