Differentiable hierarchical and surrogate gradient search for spiking neural networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Spiking neural network (SNN) has been viewed as a potential candidate for the next generation of artificial intelligence, with appealing characteristics such as sparse computation and inherent temporal dynamics. By adopting sophisticated architectures of deep artificial neural networks (ANNs) and approximating back-propagation with surrogate gradient (SG) algorithms, deep SNNs are achieving competitive performances as their artificial counterparts in benchmark machine learning tasks such as image classification. However, successful architectures of ANNs are not necessary ideal for SNN and when tasks become more diverse effective architectural variations could be critical. To this end, we develop a differentiable hierarchical search framework for spiking neurons, where spike-based computation is realized on both the cell and the layer level search space. Based on this framework, we find effective SNN architectures under limited computation cost. During the training of SNN, a suboptimal SG function could lead to poor approximations of true gradients, making the network enter certain local minima. To address this problem, we propose a differentiable surrogate gradient search method where the SG function can be efficiently optimized locally in parallel. Our models achieve state-of-the-art performances on image classification. On event-based deep stereo, our method surpasses the accuracy of specially designed ANNs with 26 $\times$ lower energy cost ($6.7\mathrm{mJ}$), demonstrating the advantage of SNN in processing highly sparse and dynamic signals.

1 Introduction

Inspired from biological neural networks, spiking neural network (SNN) [37] has been viewed as a potential candidate for the next generation of artificial intelligence, with appealing characteristics such as asynchronous computation, sparse activation and inherent temporal dynamics. However, the training of deep SNNs is challenging due to the binary spike which is incompatible with gradient-based backpropagation. To solve this problem, various surrogate gradient (SG) methods were proposed [5, 58, 42] where soft relaxed functions were used to approximate the original discontinuous gradient. Based on these methods, SNNs have achieved high level performance in benchmark deep learning tasks such as image classification on CIFAR [25] and ImageNet [12] dataset [51, 59, 65, 48, 67]. However, the accuracy of the network often drops when architectures of ANNs are directly applied to SNNs, such as ResNet [20] and VGG networks [52]. With recent improved SNN training methods [31, 15, 13] the performance gap is reducing but remains. This gap is more evident in tasks where the network architecture requires more variation, such as dense image prediction [71, 19, 23].

Directly inheriting sophisticated ANN architectures might not be ideal for SNN and there is *a lack of study* of optimal architectures for spiking neurons given a particular task. Intuitively, topology in cortex should offer us some inspiration. Neuroscience has studied macro structures such as columnar

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

organization [40] and micro dynamics such as interaction between lateral and feedback connections [32]. However, on the network level how specific connections of neural circuits relate to functionality remains largely unknown. In this work, we develop a differentiable hierarchical search method for spiking neurons in order to find optimal task-specific network architectures under limited computation cost. Based on traditional differentiable architecture search (DARTS) framework [36], we redesign the search space and information flow in the principle of spike-based computation. For both the cell and the layer search space, we study how to realize this principle towards a fully spiking network.
SG provides an approximation of the non-existing gradient in SNN and its selection is not unique. To explore optimal SG functions for the training of SNN, we propose a differentiable SG search method to efficiently adapt the function locally in parallel, and demonstrate its efficiency on both static and event-based benchmark machine learning tasks. In summary, our contributions are following:

- We develop a differentiable hierarchical search framework for spiking neurons, where spike-based computation is realized on both the cell and the layer level search space. Based on this framework, we find effective SNN architectures under limited computation cost.
- To improve gradient approximation of deep SNNs, we propose a differentiable SG search method to efficiently optimize SG functions locally in parallel, which is easy to scale and very effective for binary networks.
- Extensive experiments show that our methods surpass SNNs based on sophisticated ANN architectures on image classification of CIFAR10 and CIFAR100 datasets.
- On event-based deep stereo task, to the best of our knowledge we show the first time SNN surpasses specially designed ANNs on the Multi Vehicle Stereo Event Camera (MVSEC) [70] dataset in terms of both accuracy and network sparsity, demonstrating its advantage in processing highly sparse and dynamic signals with extremely low power and latency.

2 Related work

2.1 Architecture search

Designing high performance network architectures for specific tasks often requires expert experience and trial-and-error experiments. Neural architecture search (NAS) [14] aims to automate this manual process and has recently achieved highly competitive performance in tasks such as image classification [72, 73, 34, 49, 46], object detection [73, 8, 57, 18] and semantic segmentation [35, 66, 43, 33], etc. However, searching over a discrete set of candidate architectures often results in a massive number of potential combinations, leading to explosive computation cost. The recently proposed differentiable architecture search (DARTS) method [36] and its variations [60, 7, 10] address this problem using a continuous relaxation of the search space which enables learning a set of architecture coefficients by gradient descent, and has achieved competitive performances with the state-of-the-art using orders of magnitude fewer computation resources [36, 35, 9].

2.2 Training of SNN

The success of deep ANNs in solving benchmark machine learning tasks has motivated efforts to make SNNs realize similar capabilities, either based on bio-inspired mechanisms [45, 29, 24, 41] or approximating ANN learning algorithms [5, 58, 42, 3]. Currently, two approaches have demonstrated their efficiency, showing the ability to solve hard problems at similar levels as their artificial counterparts, namely, ANN-to-SNN conversion [50, 6, 30] and directly training SNNs with SG [51, 59, 65, 31, 15, 13]. The theoretical soundness of the SG approach for training binary activation networks has been studied and justified [4, 62]. Experiments show that the training of SNN is robust to the shape of SG function as long as it meets certain criteria, such as the overall scale [64]. [19] shows that a suitable SG function is critical when the SNN goes deeper. [31] further improved the performance of SNN by optimizing the width (or temperature) of a continuous SG function, through the guidance of an approximated gradient measured with finite difference gradient (FDG) method. However, a suitable hyperparameter setting of FDG is largely empirical. In terms of computation, it iterates over each element of the weight and is sequential across layers, making a global application of this method computationally impractical.

2.3 Event-based task with SNN

Inspired by biological retina, event camera [17] captures instantaneous changes of pixel intensity at microsecond resolution. Compared to traditional frame-based cameras, it covers a higher dynamical range (120dB) and offers a low power solution for vision tasks in high-speed scenario. Further combining it with neuromorphic processors [44, 11, 38, 16, 27] can create a fully neuromorphic system, realizing extremely low power and low latency sensing and processing. However, learning from highly sparse and asynchronous events is challenging. Given its inherited asynchronous dynamics, SNN is an ideal candidate for such task and recently a number of works have applied it for event-based problems such as classification [26, 30], tracking [61], detection [22], semantic segmentation [23] and optical flow estimation [28, 19], etc. Multi-view event-based deep stereo solves the problem of 3D scene reconstruction based on pixel differences of the same physical point from event streams obtained by multiple views. Given the problem's complexity, it has been addressed by specially designed deep ANNs. Several works use additional information such as camera motion to produce sparse depth maps [69, 68]. Estimating dense disparity images from sparse event inputs is more challenging. A recent work [55] addresses this problem by an event queue method which encodes events into event images through 3D convolution, followed by an hourglass network to estimate disparity. Based on [55], [1] further enhances local contours of the estimated disparity using image reconstruction.[39] creates feature pyramids with multi-scale correlation learned from a cycle of gray-scale images and event inputs. A very recent work [47] applies SNN to this problem by adopting a handcrafted U-net structure. Compared with ANN models using geometric volumes, the network is too simple and its performance is suboptimal.

3 Method

3.1 Spiking neuron model and SG training

We adopt the iterative leaky integrate-and-fire (LIF) neuron model [59] described by

$$u^{t,n} = \tau u^{t-1,n}(1 - y^{t-1,n}) + I^{t,n} \tag{1}$$

where superscripts n and t denote layer index and time step, respectively. τ is the membrane time constant, u is the membrane potential, y denotes the spike output and I denotes the synaptic input with $I^{t,n} = \sum_j w_j y_j^{t,n-1}$ where w is the weight. The neuron will fire a spike $y^{t,n} = 1$ when $u^{t,n}$ exceeds a threshold V_{th}, otherwise $y^{t,n} = 0$. In this work, we set $\tau = 0.2$ and $V_{th} = 0.5$. Given loss L, using chain rule the weight update of SNN can be expressed as:

$$\frac{\partial L}{\partial w} = \sum_t \frac{\partial L}{\partial y^t} \frac{\partial y^t}{\partial u^t} \frac{\partial u^t}{\partial I^t} \frac{\partial I^t}{\partial w} \tag{2}$$

where $\frac{\partial y^t}{\partial u^t}$ is the gradient of the firing function, which is zero everywhere except at $u = V_{th}$. The SG approach uses continuous functions to approximate the real gradients, such as rectangular [67], triangular [2] and exponential curves [51, 63]. We adopt Dspike function from [31]:

$$\mathrm{Dspike}(x) = a \cdot \tanh(b(x - c)) + d, \text{ if } 0 \leq x \leq 1 \tag{3}$$

with $\mathrm{Dspike}(x) = 1$ or 0 for $x > 1$ or $x < 0$. By changing the temperature b it covers a large range of choices for SG. We set $c = V_{th} = 0.5$ and determine a and d by setting $\mathrm{Dspike}(0) = 0$, $\mathrm{Dspike}(1) = 1$.

3.2 Differentiable hierarchical search for SNN

3.2.1 Cell level search

Similar to traditional DARTS [36], a cell is defined as a repeated and searchable unit, which is a directed acyclic graph with N nodes, $\{x_i\}_N$, as depicted in Fig. 1. Each cell receives input from two previous cells and forms its output by concatenating all its nodes. In DARTS-SNN, each node is a spiking neuron described by:

$$x_j = f(\sum_{i<j} o^{(i,j)}(x_i)) \tag{4}$$

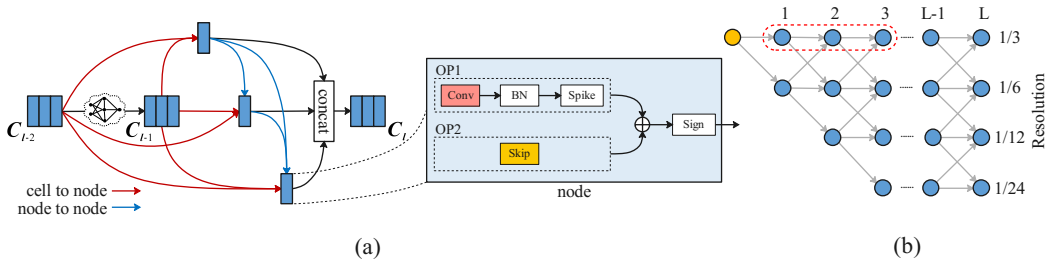


Figure 1: Hierarchical search space of DARTS-SNN. (a) Cell level search space, with 3 nodes as an example. Within one node, operations are summed (2 operations here) and filtered by a sign function. BN denotes batch normalization. (b) Layer search space for event-based deep stereo.

where f is a sign function (a weightless spiking neuron) taking the sum of all operations as input, $o^{(i,j)}$ is the operation associated with the directed edge connecting node i and j. During search, each edge is represented by a weighted average of candidate operations, the information flow connecting node i and node j becomes:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in O^{(i,j)}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o \in O^{(i,j)}} \exp(\alpha_o^{(i,j)})} o(x) \tag{5}$$

where $O^{(i,j)}$ denotes the operation space on edge (i, j) and $\alpha_o^{(i,j)}$ is the weight of operation o, which is a trainable continuous variable. At the end of search, a discrete architecture is selected by replacing each mixed operation $\bar{o}^{(i,j)}$ with the most likely operation $o^{(i,j)} = \max_{o \in O^{(i,j)}} \alpha_o^{(i,j)}$. For spiking neurons, we can either mix the operation at the spike activation or at the input of the membrane potential, i.e. $y = \bar{o}$ or $I = \bar{o}$. The former allows a search over operations with different SG functions, while the latter transfers more accurate learning signals for α and leads to a concise node with less spiking filters (see supplement material for deduction). We choose the former for search phase and apply the latter particularly for SG search during retraining when operation type freeze (Section 3.4). Specifically, in the former choice, we distinguish between operations with the same forward computation but with different SG functions in candidate operation space. In addition, in original DARTS, a prepossessing step is required on operations between nodes and cells of previous layers in order to align the dimension of feature maps. We merge this step with subsequent candidate operations to reduce model complexity and improve inference speed.

3.2.2 Layer level search

Task specific knowledge has been proved to be helpful in speeding up the search process and improving network performance. For classification task, we adopt a fixed downsampling structure (Fig. 2) as in [36], with normal cells and reduction cells searched separately. For dense image prediction where high resolution output is needed and network architecture requires more variation, we implement differentiable search on the layer level as proposed in [35]. A set of scalars $\{\beta\}$ are trained to weight different potential layer resolutions and they are updated together with α. By the end of search, an optimal structure is decoded from a pre-defined L-layer trellis, as shown in Fig. 1. For upsampling layers, we use nearest interpolation to maintain binary feature maps. For the stereo matching task, we construct a cost volume which embeds geometric knowledge of the binocular input, and search cell structure separately for the feature and matching subnetworks, similar to [9]. At the end of both subnetworks, the output of the last cell is upsampled to the initial resolution of the trellis using spike-based activation. We use batch normalization and ANN stem layers in the search phase. In the retraining phase, the ANN stem layers are converted to SNN by replacing the Relu function with spike function. Parameters of batch normalization are converted into convolution weights and biases after retraining [21, 50], leading to full spike-based network for inference. We provide more details for spike-based layer search space in the supplement.

3.3 Loss function, estimator and optimization

For classification task, we use an auxiliary loss as in [36], with weight 0.4. For event-based stereo matching, our network produces a matching cost tensor C of size $\frac{d_{max}}{2} \times h \times w$, based on which we estimate a disparity map $\hat{D}$ using a sub-pixel estimator [54].

$$\hat{D} = \sum_{d} D(d) \underset{d:|\hat{d}-d|<\delta}{\text{softmin}} (C_{d,y,x}) \text{ , with } \hat{d} = \underset{d}{\text{argmin}}(C_{d,y,x}) \tag{6}$$

where $\delta = 2$ is an estimator support and $D(d) = 2d$ is a disparity corresponding to index d in the matching cost tensor. We use a sub-pixel cross entropy loss [54] to train the network, which is described by:

$$L(\Theta) = \frac{1}{wh} \sum_{y,x} \sum_{d} \text{Laplace}(D(d)|\mu = D^{GT}_{y,x}, b) \cdot \log(\underset{d}{\text{softmin}}(C_{d,y,x})) \tag{7}$$

where $\text{Laplace}(D|\mu = D^{GT}_{y,x}, b)$ is a discretized Laplace distribution with the mean equal to the ground truth disparity $\mu = D^{GT}_{y,x}$ and diversity $b = 2$.

Following bi-level optimization [36], we update weight and architecture parameters $\{\alpha, \beta\}$ alternately based on two disjoint training sets *A* and *B*:

- Update network weights **w** by $\nabla_{\mathbf{w}} L(\mathbf{w}, \alpha, \beta)$ on *A*
- Update architecture parameters α and β by $\nabla_{\alpha,\beta} L(\mathbf{w}, \alpha, \beta)$ on *B*

We use first-order approximation to speed up the search process. After search, we decode the discrete cell structure by retaining two strongest afferent edges for each node. As to network structure, we decode it by finding the maximum probability path between different layers.

3.4 Differentiable surrogate gradient search

A recent work [19] shows that a suitable SG function is critical when the SNN goes deeper, and [31] demonstrates that by optimizing the width (or temperature) of the SG function the performance of SNN can be improved. Continuous relaxation through gradient descent is an efficient approach to explore diverse operations on the same path, inspired from this idea, we propose a differentiable surrogate gradient search (DGS) method to parallelly optimize local SGs for SNN.

In the retraining phase, with certain epoch intervals, we associate each operation path with N candidate SG functions, $\{g_i\}_N$, based on which we update the weight (or N copies of the weight) separately, leading to $\{w_{g_i}\}_N$. These weights are then combined to form a mixed operation weighted by a set of factors $\{\alpha_{g_i}\}_N$ through a softmax function, described as:

$$\hat{I} = \sum_{i=1}^{N} \frac{\exp(\alpha_{g_i})}{\sum_{j=1}^{N} \exp(\alpha_{g_j})} I_i \text{ , with } I_i = w_{g_i} x \tag{8}$$

We then update $\{\alpha_{g_i}\}_N$ through the loss of the mixed operation output. This process is repeated for multiple batches and finally we update the original SG to $\{g_i | i = \text{argmax}_i \langle \alpha_i \rangle\}$, with $\langle \cdot \rangle$ denoting the average over batches. Note that w_{g_i} can be obtained either by repeatedly calculate for each g_i, or directly estimating from the gradient of the original SG, $\nabla_{g,w} L$, if $\{g_i\}_N$ are linear to g. The psuedo code of the algorithm is summarized in Algorithm. 1. The intuition behind DGS is that the updated value of α_{g_i} indicates the contribution of w_{g_i} in decreasing the loss. So $\{g_i | i = \text{argmax}_{g_i} \langle \alpha_i \rangle\}$, which leads to the best updated weight, could be the most suitable SG function for the original local weight. Note that the difference between DGS and the SG search in Section 3.2.1 is that the former aims to optimize SG function for the local weight, while the latter is essentially a search of different operation types.

4 Experiments

4.1 Classification

4.1.1 Architecture search and retrain

We apply architecture search on CIFAR10 dataset [25] and then retrain on target datasets, including CIFAR10 and CIFAR100. Both datasets have 50K/10K training/testing RGB images with a spatial

Algorithm 1: Differentiable surrogate gradient search (DGS)

Input: Training dataset, training epoch E, training iteration I, DGS iteration I_g, SG function g, candidate SG functions $\{g_i\}_N$, SG weighting factors $\{\alpha_{g_i}\}_N$ and their initializing value ϵ, epoch interval for DGS e_D

for *all* e *= 1, 2, ..., E-th epoch* **do**
  for *all* i *= 1, 2, ..., I-iteration* **do**
    Collect training data and labels, update weights w based on SG function g;
  if $e/e_D = \text{Int.}$ **then**
    for *all* j *=1, 2, ..., I_g-iteration* **do**
      Initialize $\{\alpha_{g_i}\}_N$ to ϵ, update weights w with $\nabla_{g_i,w} L$ based on $\{g_i\}_N$, obtain associated weights $\{w_{g_i}\}_N$;
      Combine $\{w_{g_i}\}_N$ to form mixed operation, update $\{\alpha_{g_i}\}_N$ with $\nabla_{\alpha_{g_i},\{w_{g_i}\}_N} L$.
    Update g to $\{g_i | i = \text{argmax}_{g_i} \langle \alpha_i \rangle\}$
return trained network.

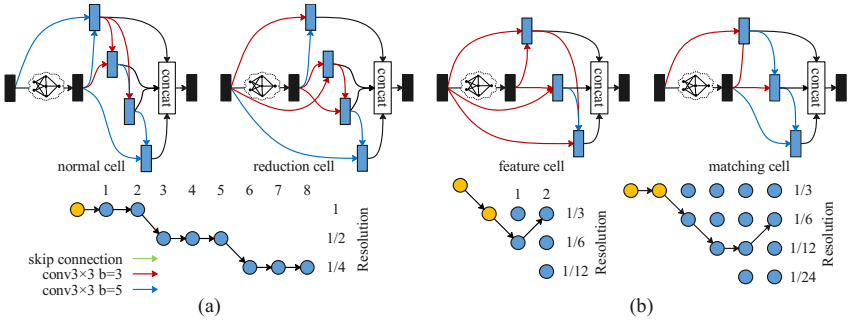


Figure 2: Searched architecture for classification (left) and event-based stereo matching (right).

resolution of 32×32. In the search phase, the training set is equally split into two subsets for bi-level optimazation. For retraining, the standard training/testing split is used. We use 4 nodes within one cell and a limited number of candidate operations to reduce search time, which are {conv3×3 with $g(b = 3)$, conv3×3 with $g(b = 5)$, skip connection}, with $g = \text{Dspike}$. For the sign function we use $g(b = 3)$. For the network architecture, we adopt an 8 layer fixed downsampling architecture proposed in [36]. The search phase takes 50 epochs with mini-batch size 50, the first 15 epochs are used to warm up convolution weights. We use SGD optimizer with momentum 0.9 and a learning rate of 0.025. The architecture search takes about 1.4 GPU day on a single NVIDIA Tesla V100 (32G) GPU. After search, we retrain the model on target datasets with channel expansion for 100 epochs with mini-batch size 50 and cosine learning rate 0.025. We use SGD optimizer with weight decay $3e^{-4}$ and momentum 0.9. The DGS method is applied to the first stem layer, we set $I_g = 100$, $\epsilon = 0.001$, $e_D = 5$, $g = \text{Dspike}$ with $\{g_i\}_N$ having equal temperature interval: $\{g(b - \frac{(N-1)}{2}\Delta b), ..., g(b + \frac{(N-1)}{2}\Delta b)\}$ where $\Delta b = 0.2$ and $N = 5$. More details about the architecture and training can be found in the supplement material.

4.1.2 Results

The results are summarized in Table 1 and the values of other models are obtained from their papers. DARTS-SNN trained with DGS achieves the highest accuracy on both datasets. Note that both Dspike [31] and TET [13] use advanced training algorithms while plain DARTS-SNN are trained with fixed SG function. Our model also exhibits highly sparse activation, as shown in Fig. 5. As a reference, we run ANN DARTS with the same layer depth and achieve 95.88% accuracy on CIFAR10. The searched architecture is shown in Fig. 2. We conjecture that the large number of jump connections both within and across cells potentially help gradient propagation through deep layers, which improves training of the model.

Table 1: Comparison on image classification. ^{D}: DGS method.

Dataset	Model	Methods	Architecture	Simulation Length	Accuracy[%]
CIFAR10	[65]	TSSL-BP	CIFARNet	5	91.41
	[48]	Diet-SNN	ResNet-20	10	92.54
	[67]	STBP-tdBN	ResNet-19	6	93.16
	[31]	Dspike	ResNet-18	6	94.25 ± 0.07
	[13]	TET	ResNet-19	6	94.50 ± 0.07
	our model	DARTS-SNN	DARTS-SNN	6	94.34 ± 0.06
	our model	DARTS-SNN^{D}	DARTS-SNN^{D}	6	$\mathbf{94.68 \pm 0.05}$
CIFAR100	[48]	Diet-SNN	ResNet-20	5	64.07
	[67]	STBP-tdBN	ResNet-19	6	71.12 ± 0.57
	[31]	Dspike	ResNet-18	6	74.24 ± 0.10
	[13]	TET	ResNet-19	6	74.72 ± 0.28
	our model	DARTS-SNN	DARTS-SNN	6	75.70 ± 0.14
	our model	DARTS-SNN^{D}	DARTS-SNN^{D}	6	$\mathbf{76.03 \pm 0.20}$

4.2 Event-based deep stereo

We further apply our method to event-based deep stereo matching on the widely used benchmark MVSEC dataset [70]. The dataset contains depth information recorded by LIDAR sensors and event streams collected from a pair of *Davis346* cameras, with synchronized 20 Hz gray scale images at 346×260 resolution. We split and preprocess the Indoor Flying dataset from the MVSEC following the same setting as [55, 1, 69]. We use the mean depth error (MDE), one-pixel-accuracy (1PA), median depth error, and mean disparity error as evaluation metrics. Learning from highly sparse raw events is challenging and a prepossessing step is often required to encode events. We use stacking based on time (SBT) [56] which merges events into temporally neighboring frames. During training, we use multiple consecutive stacks as one input, with an equal number of consecutive ground truth disparities as one label. More details can be found in the supplement material.

4.2.1 Architecture search and retrain

We use 3 nodes within one cell. For candidate operations, we use: {conv3×3 with $g(b = 3)$, conv3×3 with $g(b = 5)$, skip connection} for the feature net and {conv3×3×3 with $g(b = 3)$, conv3×3×3 with $g(b = 5)$, skip connection} for the matching net, with $g = \mathrm{Dspike}$. For the layer search space, we adopt a four-level trellis with downsampling rates of {3,2,2,2}. The number of layers is set to 2 and 4 for the feature and matching net, respectively. In addition, two stem layers are applied in front of both subnetworks to reduce input spatial resolution and increase channel number. We search the architecture for 12 epochs with batch size 1. The first 3 epochs are used to initiate the weight of the supernet. The rest 9 epochs are applied with bi-level optimization. We use SGD optimizer with momentum 0.9 and a learning rate of 0.002. The architecture search takes about 0.4 GPU day on a single NVIDIA Tesla V100 (32G) GPU. The searched architecture is shown in Fig. 2. Extensive random seed experiments show that the architectures are gradually optimized during the search phase. After search, we retrain the model with channel expansion for 200 epochs with mini-batch size 2. We use Adam optimizer with initial learning rate 0.001 and momentum $(0.9, 0.999)$, with learning rate decaying half at $[50, 100, 150]$ epochs. The DGS method is applied to the first stem layer of the feature net with the same setup as in classification. We provide more details in the supplement.

4.2.2 Results

We compare our method with the former described state-of-the-art event-based stereo matching approaches (Section 2.3) on dense disparity estimation. The results are summarized in Table 2. Values of other models are obtained from their papers. Among SNNs methods, DARTS-SNN significantly outperforms StereoSpike, exhibiting the effectiveness of the searched architecture. Among events-only approaches, DARTS-SNN even surpasses ANN based specially designed DDES network in all criteria with only about one-third of the number of parameters. The performance of DARTS-SNN/BN are further improved with DGS. As shown in Fig. 4, the temperature of the SG function is constantly optimized, which avoids vanishing gradient compared to training with fixed SG function, leading to more stable training of SNN. We also make a qualitative comparison of estimated disparities in

Table 2: We denote the best and second best results in bold and underscore. EO denotes events-only method. EITNet requires gray scale images for training but not for inference. Symbols meaning: -, unavailability of the value; (·), streaming tests; ^{D}, training with DGS; *, estimated value.

Method	EO	No. param.	MDE [cm] ↓		Median depth error [cm] ↓		Mean disparity error [pix] ↓		1PA [%] ↑	
			split1	split3	split1	split3	split1	split3	split1	split3
EIS [39]	✗	-	**13.7**	22.4	-	-	-	-	89.0	88.1
EITNet [1]	✓̸	>16M*	<u>14.2</u>	<u>19.4</u>	**5.9**	10.4	<u>0.55</u>	0.75	**92.1**	**89.6**
DDES [55]	✓	2.33M	16.7	27.8	6.8	14.7	0.59	0.94	89.4	74.8
StereoSpike [47]	✓	-	18.5	25.4	-	-	-	-	-	-
DARTS-BN	✓	0.87M	17.5	20.3	7.1	10.8	0.59	0.74	88.7	88.3
DARTS-BN^{D}	✓	0.87M	17.0	19.8	6.8	**10.2**	0.58	0.73	89.3	88.5
DARTS-SNN	✓	0.87M	16.5(16.5)	19.4(19.5)	6.5(6.5)	10.6(10.6)	0.57(0.57)	<u>0.73</u>(0.74)	90.1(90.2)	88.5(88.4)
DARTS-SNN^{D}	✓	0.87M	15.9(15.8)	**19.1**(19.3)	<u>6.3</u>(6.3)	<u>10.4</u>(10.5)	**0.54**(0.54)	**0.72**(0.74)	<u>90.7</u>(90.8)	<u>88.9</u>(88.8)

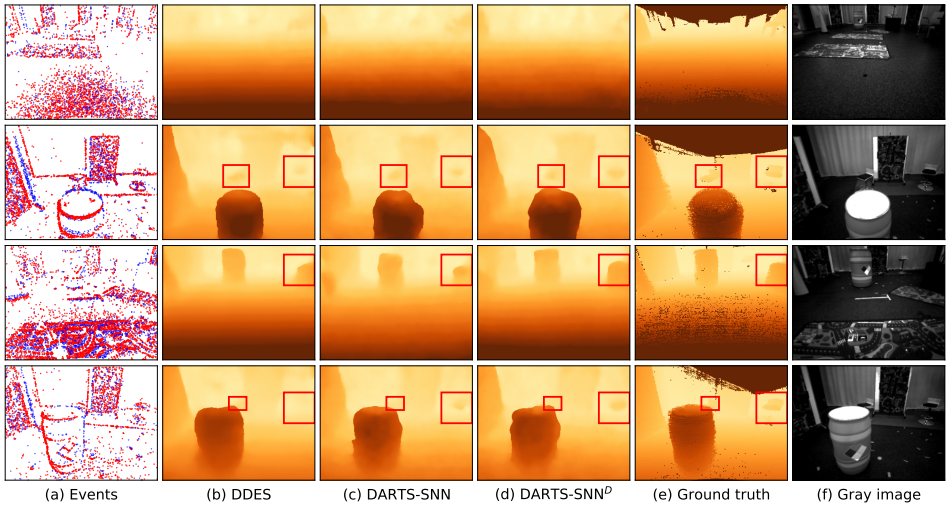


Figure 3: Qualitative comparison on MVSEC. Disparity maps of different methods are on same frames, see supplement material for details.

Fig. 3. It can be seen that DARTS-SNN with DGS predicts the best disparities among all the models, especially in local edges.

Streaming inference: In real-world scenario, events are generated consecutively by the sensor with flexible lengths. To test the real-time applicability of our model, we fed the entire test split continuously into the model, which evolves an equal length of steps and estimates sequential disparities. The inference speed of our model achieves 44 FPS (26 FPS for the DDES model, see supplement material for measurement details) while achieving similar accuracy as in standard testing (Fig. 3), where the model always receives a fixed length of events.

4.2.3 Ablation study

The inherent temporal dynamics of SNN is assumed to be helpful for it to learn temporal correlation of the data. To investigate this property, we fix the membrane time constant τ to 0 to create binary neuron (BN) and perform architecture search and retrain. As shown in Table 2, the DARTS-BN model performs worse than DARTS-SNN, which shows the benefit of temporal dynamics in this task.

5 Sparsity and energy cost

The dense computation of deep ANNs came at a significant energy cost. In contrast, SNN performs sparse computing and multiplication-free inference. As Fig. 5 shows, our models show an overall

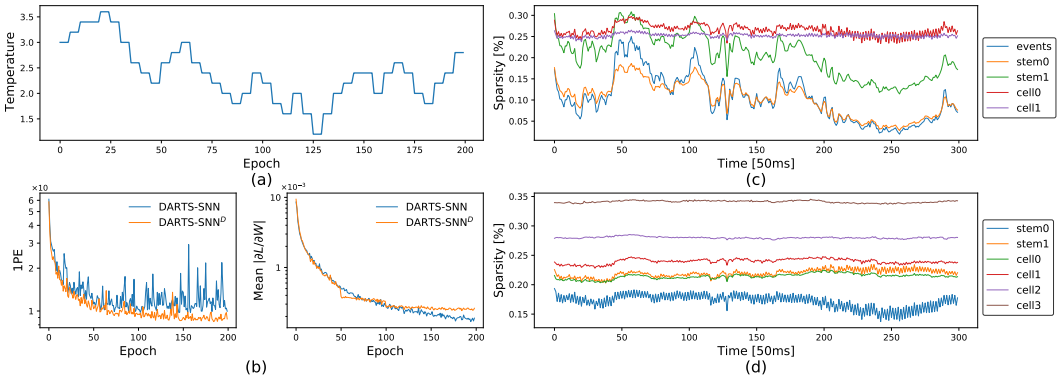
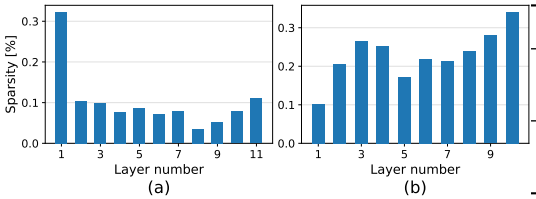


Figure 4: DGS training process and sparsity of DARTS-SNN. (a): Evolution of SG temperatures during DGS training. (b): 1PE (left) and first stem weight gradient (right) of DGS training and normal training. (c) and (d): Sparsity of the feature net and the matching net.

sparse activity with different degrees across layers. For event-based stereo, we plot network activity along with the density of events for a short duration (Fig. 4). The activity of the first stem layer in the feature net highly correlates with the density of the event streams. This relation is weakened with the increase of layer depth. As activity propagates, the sparsity of matching layers drops to create dense disparities. The operation number of ANN is collected by a public available PyTorch package [53]. Following [31], the addition count of SNN is calculated by $s * T * A$, where s is the mean sparsity, T is the time step and A is the addition number. The operation number and energy cost is in Table 3. We can see that DARTS-SNN has much lower operation number than DDES. Note that due to streaming inference, SNN realizes a natural usage of its temporal dynamics with $T = 1$. We measure the energy consumption following [48]. In 45nm CMOS technology, the addition operation in SNN costs 0.9pJ while the multiply-accumulate (MAC) operation in ANN consumes 4.6pJ. DARTS-SNN costs only 6.7mJ for a single forward, with $26\times$ lower energy than ANN.



Method	#Add.	#Mult.	Energy
DARTS CIFAR10	29654M	29654M	136.4 mJ
DARTS-SNN CIFAR10	**5973M**	**19M**	**5.5** mJ
DDES (ANN)	37917M	37917M	174.4 mJ
DARTS-SNN MVSEC	**7414M**	**0M**	**6.7** mJ

Figure 5 & Table 3: Left: Network sparsity for (a) classification and (b) event-based stereo. Right: The operation number and energy cost.

6 Discussion

In this paper, we develop a differentiable hierarchical search framework for spiking neurons which finds effective architectures for sparse computation, achieving state-of-the-art performances on image classification and event-based deep stereo. The latter demonstrates the advantage of SNNs in processing highly sparse and dynamic event streams with extremely low power. One potential reason of the high performance of our architecture is the large number of jump connections both within and across cells, which helps gradient propagation. Our method can be applied to other tasks where SNN requires more architectural variations such as object detection and semantic segmentation. The DGS algorithm efficiently optimizes SGs in a distributed manner, which can be used as a general method to improve the training of deep SNNs or binary networks. Currently we only apply DGS on fixed layer, a potential better solution could be a dynamical algorithm to apply it on optimal layers. Another limitation is that we only apply our method on the simple iterative LIF neuron, future works can explore an optimal combination of different types of neurons. In addition, we only use limited candidate operations, other types of biological plausible connections such as recurrent and feedback connections, as well as excitatory/inhibitory synapses could be considered in future work.

References

[1] Soikat Hasan Ahmed, Hae Woong Jang, SM Nadim Uddin, and Yong Ju Jung. Deep event stereo leveraged by event-to-image translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 882–890, 2021.

[2] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in neural information processing systems*, 31, 2018.

[3] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):1–15, 2020.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[5] Sander M Bohte, Joost N Kok, and Johannes A La Poutré. Spikeprop: backpropagation for networks of spiking neurons. In *ESANN*, volume 48, pages 419–424. Bruges, 2000.

[6] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2021.

[7] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1294–1303, 2019.

[8] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. *Advances in Neural Information Processing Systems*, 32, 2019.

[9] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Hongdong Li, Tom Drummond, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. *Advances in Neural Information Processing Systems*, 33:22158–22169, 2020.

[10] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. Darts-: robustly stepping out of performance collapse without indicators. *arXiv preprint arXiv:2009.01027*, 2020.

[11] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R Risbud. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

[14] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[15] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[16] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[17] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1): 154–180, 2020.

[18] Jianyuan Guo, Kai Han, Yunhe Wang, Chao Zhang, Zhaohui Yang, Han Wu, Xinghao Chen, and Chang Xu. Hit-detector: Hierarchical trinity architecture search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11405–11414, 2020.

[19] Jesse Hagenaars, Federico Paredes-Vallés, and Guido De Croon. Self-supervised learning of event-based optical flow with spiking neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[22] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: Spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11270–11277, 2020.

[23] Youngeun Kim, Joshua Chough, and Priyadarshini Panda. Beyond classification: Directly training spiking neural networks for semantic segmentation. *arXiv preprint arXiv:2110.07742*, 2021.

[24] Agnes Korcsak-Gorzo, Michael G Müller, Andreas Baumbach, Luziwei Leng, Oliver J Breitwieser, Sacha J van Albada, Walter Senn, Karlheinz Meier, Robert Legenstein, and Mihai A Petrovici. Cortical oscillations support sampling-based computations in spiking neural networks. *PLoS computational biology*, 18(3):e1009753, 2022.

[25] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

[26] Alexander Kugele, Thomas Pfeil, Michael Pfeiffer, and Elisabetta Chicca. Efficient processing of spatio-temporal data streams with spiking neural networks. *Frontiers in neuroscience*, 14: 439, 2020.

[27] Akos F Kungl, Sebastian Schmitt, Johann Klähn, Paul Müller, Andreas Baumbach, Dominik Dold, Alexander Kugele, Eric Müller, Christoph Koke, Mitja Kleider, et al. Accelerated physical emulation of bayesian inference in spiking neural networks. *Frontiers in neuroscience*, page 1201, 2019.

[28] Chankyu Lee, Adarsh Kumar Kosta, Alex Zihao Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks. In *European Conference on Computer Vision*, pages 366–382. Springer, 2020.

[29] Luziwei Leng, Roman Martel, Oliver Breitwieser, Ilja Bytschok, Walter Senn, Johannes Schemmel, Karlheinz Meier, and Mihai A Petrovici. Spiking neurons with short-term synaptic plasticity form superior generative networks. *Scientific reports*, 8(1):1–11, 2018.

[30] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, pages 6316–6325. PMLR, 2021.

[31] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[32] Hualou Liang, Xiajing Gong, Minggui Chen, Yin Yan, Wu Li, and Charles D Gilbert. Interactions between feedback and lateral connections in the primary visual cortex. *Proceedings of the National Academy of Sciences*, 114(32):8637–8642, 2017.

[33] Peiwen Lin, Peng Sun, Guangliang Cheng, Sirui Xie, Xi Li, and Jianping Shi. Graph-guided architecture search for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4203–4212, 2020.

[34] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34, 2018.

[35] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 82–92, 2019.

[36] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[37] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[38] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345 (6197):668–673, 2014.

[39] Mohammad Mostafavi, Kuk-Jin Yoon, and Jonghyun Choi. Event-intensity stereo: Estimating depth by the best of both worlds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4258–4267, 2021.

[40] Vernon B Mountcastle. The columnar organization of the neocortex. *Brain: a journal of neurology*, 120(4):701–722, 1997.

[41] Emre Neftci, Srinjoy Das, Bruno Pedroni, Kenneth Kreutz-Delgado, and Gert Cauwenberghs. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in neuroscience*, 7:272, 2014.

[42] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[43] Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9126–9135, 2019.

[44] J. Pei, L. Deng, S. Song, M. Zhao, and L. Shi. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106, 2019.

[45] Mihai A Petrovici, Johannes Bill, Ilja Bytschok, Johannes Schemmel, and Karlheinz Meier. Stochastic inference with spiking neurons in the high-conductance state. *Physical Review E*, 94 (4):042312, 2016.

[46] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.

[47] Ulysse Rançon, Javier Cuadrado-Anibarro, Benoit R Cottereau, and Timothée Masquelier. Stereospike: Depth learning with a spiking neural network. *arXiv preprint arXiv:2109.13751*, 2021.

[48] Nitin Rathi and Kaushik Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*, 2020.

[49] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[50] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

[51] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *arXiv preprint arXiv:1810.08646*, 2018.

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[53] Vladislav Sovrasov. Flops counter for convolutional networks in pytorch framework, 2019. URL `https://github.com/sovrasov/flops-counter.pytorch/`.

[54] Stepan Tulyakov, Anton Ivanov, and Francois Fleuret. Practical deep stereo (pds): Toward applications-friendly deep stereo matching. *arXiv preprint arXiv:1806.01677*, 2018.

[55] 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*, pages 1527–1537, 2019.

[56] Lin Wang, Yo-Sung Ho, Kuk-Jin Yoon, et al. Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10081–10090, 2019.

[57] Ning Wang, Yang Gao, Hao Chen, Peng Wang, Zhi Tian, Chunhua Shen, and Yanning Zhang. Nas-fcos: Fast neural architecture search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11943–11951, 2020.

[58] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[59] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1311–1318, 2019.

[60] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

[61] Zheyu Yang, Yujie Wu, Guanrui Wang, Yukuan Yang, Guoqi Li, Lei Deng, Jun Zhu, and Luping Shi. Dashnet: A hybrid artificial and spiking neural network for high-speed object tracking. *arXiv preprint arXiv:1909.12942*, 2019.

[62] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.

[63] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[64] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Computation*, 33(4):899–925, 2021.

[65] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in Neural Information Processing Systems*, 33:12022–12033, 2020.

[66] Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. Customizable architecture search for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11641–11650, 2019.

[67] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11062–11070, 2021.

[68] Yi Zhou, Guillermo Gallego, Henri Rebecq, Laurent Kneip, Hongdong Li, and Davide Scaramuzza. Semi-dense 3d reconstruction with a stereo event camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 235–251, 2018.

[69] Alex Zihao Zhu, Yibo Chen, and Kostas Daniilidis. Realtime time synchronized event-based stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 433–447, 2018.

[70] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018.

[71] Lin Zhu, Xiao Wang, Yi Chang, Jianing Li, Tiejun Huang, and Yonghong Tian. Event-based video reconstruction via potential-assisted spiking neural network. *arXiv preprint arXiv:2201.10943*, 2022.

[72] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[73] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract and Introduction.
   (b) Did you describe the limitations of your work? [Yes] See Discussion.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is concerned with technical basic research and we do not foresee any such direct negative impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] It does.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See both the main text and the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See tables in Section 4.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.1.1 and Section 4.2.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Public resource.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]