

# Neural Architecture Search for Spiking Neural Networks

Youngeun Kim, Yuhang Li, Hyungseob Park, Yeshwanth Venkatesha, and Priyadarshini Panda  
 Department of Electrical Engineering  
 Yale University  
 New Haven, CT, USA

{youngeun.kim, yuhang.li, hyungseob.park, yeshwanth.venkatesha, priya.panda}@yale.edu

## Abstract

Spiking Neural Networks (SNNs) have gained huge attention as a potential energy-efficient alternative to conventional Artificial Neural Networks (ANNs) due to their inherent high-sparsity activation. However, most prior SNN methods use ANN-like architectures (e.g., VGG-Net or ResNet), which could provide sub-optimal performance for temporal sequence processing of binary information in SNNs. To address this, in this paper, we introduce a novel Neural Architecture Search (NAS) approach for finding better SNN architectures. Inspired by recent NAS approaches that find the optimal architecture from activation patterns at initialization, we select the architecture that can represent diverse spike activation patterns across different data samples without training. Moreover, to further leverage the temporal information among the spikes, we search for feed forward connections as well as backward connections (i.e., temporal feedback connections) between layers. Interestingly, SNASNet found by our search algorithm achieves higher performance with backward connections, demonstrating the importance of designing SNN architecture for suitably using temporal information. We conduct extensive experiments on three image recognition benchmarks where we show that SNASNet achieves state-of-the-art performance with significantly lower timesteps (5 timesteps). We provide the code in Supplementary.

## 1. Introduction

Spiking Neural Networks (SNNs) [14, 24, 48, 71, 84, 85] have gained increasing attention as a promising paradigm for low-power intelligence. Inspired by biological neuronal functionality, SNNs process visual information with binary spikes over multiple timesteps. Majority of works on SNNs have so far focused on image classification problem [71] to develop an energy-efficient alternative to Artificial Neural Networks (ANNs). To this end, recent SNN works utilize ANN architectures (e.g., VGG-Net [77] or ResNet [35]) de-

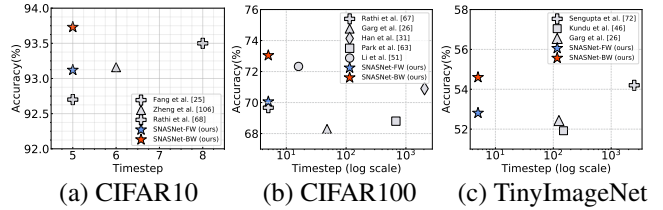


Figure 1. Accuracy and timesteps for different SNN models on (a) CIFAR10 (b) CIFAR100 and (c) TinyImageNet dataset. While showing comparable accuracy with state-of-the-art networks, SNASNet achieves significantly lower latency. Also, SNASNet-BW where we search both forward and backward connections provides better performance than the SNASNet-Fw with only forward connections.

signed by human experts. While SNNs show an impressive advantage on energy-efficiency, SNNs still lag behind in terms of accuracy compared to ANNs.

In this paper, we assert that the inherent structural/functional difference between ANNs and SNNs induces an unignorable architectural gap, resulting in a sub-optimal solution when we naively deploy ANN architectures on SNNs. Specifically, different from ANNs with ReLU neurons, SNNs consist of Leaky Integrate-and-Fire (LIF) neurons which store and transmit temporal information. However, manually searching for SNN-friendly architectures is laborious. Therefore, we use Neural Architecture Search (NAS) [8, 37, 75, 102, 108, 109], which can automatically discover the optimal SNN architecture. Although NAS has become a prevalent technique in various ANN tasks [11, 13, 27, 104], NAS for designing SNNs has not been investigated. In this work, we ask two questions:

Q1. Which NAS algorithm is suitable for SNNs?

Q2. Which SNN architecture provides better performance on an image recognition task?

For the first question, we highlight that the mainstream NAS algorithms either require multiple training stages [2, 78, 107–109] or require training a supernet once with all architecture candidates [8, 29, 55, 86] which takes longer training time to converge than standard training. As SNNs have

a significantly slower training process compared to ANNs (*e.g.*, training SNN on MNIST with NVIDIA V100 GPU takes  $11.43\times$  more latency compared to the same ANN architecture [53]), the above NAS approaches are difficult to be applied on SNNs. On the other hand, recent works [12, 58, 88] have proposed efficient NAS approaches that search the best neuron cell from initialized networks without any training. Specifically, [58] shows that the network architecture with a high representation power at initialization is likely to achieve higher post-training accuracy. Motivated by this, without the training process, we select the SNN architecture that can represent diverse spike activation patterns across different data samples. To quantify the diversity of networks, we measure the distance of temporal activation patterns between different mini-batch samples. However, SNNs show high sparsity variation on the temporal patterns across different mini-batches, resulting in inaccurate distance measures. To address this, we normalize the distance measure based on the sparsity of given activation patterns, that we term as Sparsity-Aware Hamming Distance (SAHD).

To answer the second question, we search the optimal architecture block for SNNs. Here, we find the connection topology as well as the corresponding operation for each connection following previous works [21, 96]. Different from ANNs, SNNs can leverage backward connections as they convey information through time. The backward connections in SNNs can compute more efficiently because each neuron can participate several times in a network computation [3], and they are likely to capture the temporal correlation of the given input. A line of work has studied backward connections in SNNs with various architectures and training methods [3, 16, 40, 62, 63, 100]. Therefore, we search backward connections as well as forward connections through our NAS algorithm. Surprisingly, SNNs with backward connections yield improved accuracy by up to 3% across various benchmark datasets compared to SNNs with forward connections only. Also, as shown in Fig. 1, *SNASNet* founded by our NAS algorithm achieves state-of-the-art performance with significantly small number of timesteps.

In summary, our key contributions are as follows: (i) So far, most SNN literature deploys architectures from ANN models which can yield sub-optimal performance for SNNs. For the first time, we showcase a NAS technique for finding better SNN architecture on the image recognition task. (ii) Motivated by the prior work [12, 58, 88], we find an SNN-friendly architecture by comparing temporal activation without any training process. Eliminating the training cost to find the optimal architecture brings a huge advantage for SNNs that require significantly longer training time compared to ANNs. (iii) We also propose Sparsity-Aware Hamming Distance (SAHD) for addressing sparsity variation of LIF neurons. (iv) Furthermore, we search backward connections for leveraging temporal correlation in spiking in-

puts, which has not been explored before in NAS approaches for ANN architecture.

## 2. Related Work

### 2.1. Spiking Neural Networks

Spiking Neural Networks (SNNs) have gained great attention as an energy-efficient alternative over standard Artificial Neural Networks (ANNs) [9, 15, 19, 20, 31, 41–45, 48, 48, 52, 60, 62, 71–73, 79, 95, 97]. SNNs process temporal information through weight connections and a Leak-Integrate-and-Fire (LIF) neuron [39] which works as a non-linear activation in SNNs. The LIF neuron has its own memory called membrane potential that can store the temporal spike dynamics by accumulating incoming spike signals. If the membrane potential exceeds a firing threshold, the neuron generates a post-synaptic spike. The integrate-and-fire behavior of neurons induces non-differentiable transfer function. As a result, standard backpropagation is difficult to be applied during the training phase [61]. Therefore, various methods have been proposed to circumvent the non-differentiable backpropagation problem. Spike-timing-dependent plasticity (STDP) reinforces or punishes the neuronal connection based on the spike history [5, 6, 36, 41, 90, 97]. Also, a line of work [20, 30, 31, 52, 72, 73, 98] approximate ReLU with LIF by converting pre-trained ANNs to SNNs using weight or threshold balancing.

Recently, surrogate gradient learning approaches have become popular [28, 50, 51, 61, 74, 84] due to their higher performance and smaller number of timesteps compared to other training techniques. They define a surrogate function for LIF neurons when calculating backward gradients, thus the non-differentiable problem of LIF neurons is resolved. Wu *et al.* [85] represent the LIF model in a discrete-time domain and enable SNN training with a Pytorch platform. The authors of [81] propose a training algorithm that calculates backward gradients of the accumulated input and output spikes over the time window. Tandem learning [82, 83] utilizes an auxiliary ANN that facilitates stable error backpropagation for SNN training. A line of work [25, 68] train membrane decay or firing threshold in an LIF neuron, which improves the representation power of SNNs. Also, Batch Norm (BN) [38] has been applied to accelerate the training process of SNNs [42, 49, 106]. In spite of the recent developments in SNN training techniques, all of the prior methods leverage ANN architecture, such as, VGG and ResNet families. We assert that these architectures may provide sub-optimal solution for SNNs. Different from previous methods, we search better SNN architectures for the image recognition task which has not been explored so far.

### 2.2. Neural Architecture Search

Neural Architecture Search (NAS) has been proposed to discover high-performance networks [8, 37, 75, 102, 108, 109].

The early stage of NAS algorithm uses reinforcement learning [2, 78, 107–109] or evolutionary algorithm [70]. However, such methods require training the searched architecture from scratch for each search step, which is extremely computationally expensive. To address this, weight-sharing approaches have been proposed [4, 7, 8, 10, 29, 55, 66, 80, 86, 93, 103]. They train the supernet once which includes all architecture candidates. For instance, Darts [55] jointly optimizes the network parameters and the importance of each architecture candidate. Also, SPOS [29] trains the weight parameters with uniform forward path sampling and finds the optimal architecture via evolutionary strategy. The weight-sharing methods do not require training the architecture from scratch at each search step, resulting in better efficiency compared to previous NAS algorithms. In very recent works, the key focus has been the efficiency of the NAS technique [1, 91, 92, 94, 105] owing to the growing size of dataset and architecture. Interestingly, a line of work suggests the concept of NAS without training where the networks do not require training during the search stage [12, 58, 88]. This can significantly reduce the computational cost for searching optimal architecture. At the same time, several benchmarks [21, 22, 76, 96] have been proposed in order to remove the burden of training time. Following the success of NAS on image classification domain, NAS has been deployed on various tasks such as object detection [13], segmentation [54, 104], GAN [27], transformer [11], and human pose estimation [89, 99]. Despite the huge progress of NAS algorithm in ANN domain, NAS for SNNs has not been developed yet. In this work, we aim to build better SNN architecture by leveraging NAS. Different from the previous methods that search only forward connections of the networks, we search for backward connections in addition to forward, which enhances the temporal correlation of spikes.

### 3. Preliminaries

#### 3.1. Leaky Integrate-and-Fire neuron

Leaky Integrate-and-Fire (LIF) neuron is widely used for constructing SNNs [25, 71, 85]. A neuron has a membrane potential  $U_m$  that stores the temporal spike information, which can be represented as follows:

$$\tau_m \frac{dU_m}{dt} = -U_m + I(t). \quad (1)$$

Here,  $I(t)$  is an input signal for the LIF neuron, and  $\tau_m$  is a time constant for decaying the membrane potential. We convert the above continuous differential equation into a discrete version as in previous works [25, 85]:

$$u_i^t = (1 - \frac{1}{\tau_m})u_i^{t-1} + \frac{1}{\tau_m} \sum_j w_{ij} o_j^t, \quad (2)$$

where,  $t$  denotes the timestep,  $u_i^t$  represents the membrane potential of a neuron  $i$ . Also,  $w_{ij}$  stands for weight connections

between neuron  $i$  and neuron  $j$ . The neuron  $i$  accumulates membrane potential and generates a spike output  $o_i^t$  whenever membrane potential exceeds the threshold  $\theta$ :

$$o_i^t = \begin{cases} 1, & \text{if } u_i^t > \theta, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

The membrane potential is reset to zero after firing.

#### 3.2. NAS without Training

Compared to standard ANNs, SNNs require significantly higher computational cost for training due to multiple feed-forward steps [53]. This makes it difficult to search for an optimal SNN architecture with NAS techniques that train the architecture candidate multiple times [2, 78, 107–109] or train a complex supernet [8, 29, 55, 86]. To minimize the training budget, our work is motivated by the previous works [12, 58, 88] which demonstrate that the optimal architecture can be founded without any training process. Specifically, Mellor *et al.* [58] provide the interesting observation that the architecture having distinctive representations across different data samples is likely to achieve higher post-training performance. To measure discriminative power of initialized networks, they utilize the activation pattern of ReLU neurons as a binary indicator. If the ReLU neuron generates a positive value (*i.e.*, input  $> 0$ ), the neuron is mapped to 1; otherwise 0. As a result, ReLU neurons in one layer can be encoded to binary vector  $\mathbf{c}$ . Given  $N$  samples in one mini-batch, they construct a kernel matrix by computing Hamming distance  $d_H(\mathbf{c}_i, \mathbf{c}_j)$  between different samples  $i$  and  $j$ , which can be formulated as follows:

$$\mathbf{K}_H = \begin{pmatrix} N_A - d_H(\mathbf{c}_1, \mathbf{c}_1) & \cdots & N_A - d_H(\mathbf{c}_1, \mathbf{c}_N) \\ \vdots & \ddots & \vdots \\ N_A - d_H(\mathbf{c}_N, \mathbf{c}_1) & \cdots & N_A - d_H(\mathbf{c}_N, \mathbf{c}_N) \end{pmatrix} \quad (4)$$

Here,  $N_A$  stands for the number of ReLU neurons in the given layer. The final score of the architecture candidate is obtained by:

$$s = \log(\det | \sum_l \mathbf{K}_H^l |), \quad (5)$$

where,  $\mathbf{K}_H^l$  is the kernel matrix at layer  $l$ . High score implies low off-diagonal elements of kernel matrix  $\mathbf{K}_H$ , which means that the activation patterns from difference samples are not similar. Finally, the highest-scored architecture among the candidates is selected for training.

### 4. Methodology

In this section, we first introduce a temporal binary indicator of an LIF neuron based on the concept of linear region in neural networks. After that, we present sparsity-aware hamming distance that accounts for the sparsity variation of an LIF neuron. Finally, we provide the search space for our NAS algorithm where we find both forward and backward connections.

#### 4.1. Linear Regions from LIF neurons

NAS without training approaches in ANN domain [12, 58] are based on the theoretical concept of linear region in neural networks [32, 33, 59, 67, 87]. That is, each piecewise linear function (such as, ReLU) divides the input space into multiple linear regions. The composition of multiple piecewise linear functions brings multiple linear regions on the input space. Such pattern of linear regions is used for measuring the representation power of initialized networks by comparing the patterns between different samples. Here, based on previous work, we introduce the definition of neuron transition (*i.e.*, the boundary of linear region) in a piecewise linear function.

**Definition 1.** (Raghu *et al.* [67]) *For fixed  $W$ , we say a neuron with piecewise linear region transitions between inputs  $x$ ,  $x + \delta$  if its activation function switches linear region between  $x$  and  $x + \delta$ .*

For instance, ReLU and Hard Tanh have neuron transition at 0 and  $\{-1, 1\}$ , respectively [67]. Fig. 2(a) also shows the simple example with three ReLU neurons. The input space is divided into two regions by a single ReLU neuron. By composing ReLU neurons, the input space is partitioned into multiple regions where each region represents different linear function.

According to Definition 1, a LIF neuron can be regarded as piecewise linear function. For each timestep, the LIF neuron transfers 0 if the membrane potential is lower than a firing threshold, otherwise it generates 1 (*i.e.*, spike). Thus, neuron transition occurs when a given input generates an output spike. We illustrate the transfer function of an LIF neuron in Fig. 2(b). Different from ReLU neuron, the output of LIF neuron is not solely dependent on the input. As we shown in Eq. 2 and Eq. 3, the output of LIF neuron is based on the current input as well as the previous membrane potential. Therefore, the neuron transition point can be changing across time. For example, suppose that the firing threshold is 1 and the membrane potential from previous timestep is 0.3. In this case, neuron transition happens at  $input = 0.7$ . After the neuron fires, the membrane potential is reset to 0, where, the neuron transition point becomes 1. With this time-varying transfer function, the linear region of SNNs become more diverse.

#### 4.2. Sparsity-Aware Hamming Distance

In NASWOT [58], Hamming Distance (HD) is a key metric to compare the binary activation pattern  $c_i, c_j$  between two different mini-batch samples  $i, j$ . However, standard HD gives inaccurate distance measurement for SNNs due to the large sparsity variance of binary activation pattern  $c$  of LIF. Here, the term “sparsity” denotes the percentage of 0 in binary activation pattern  $c$  from one layer at a given timestep  $t$ . Note, the definition of “sparsity” here is slightly different from the previous works which defines “sparsity” from the

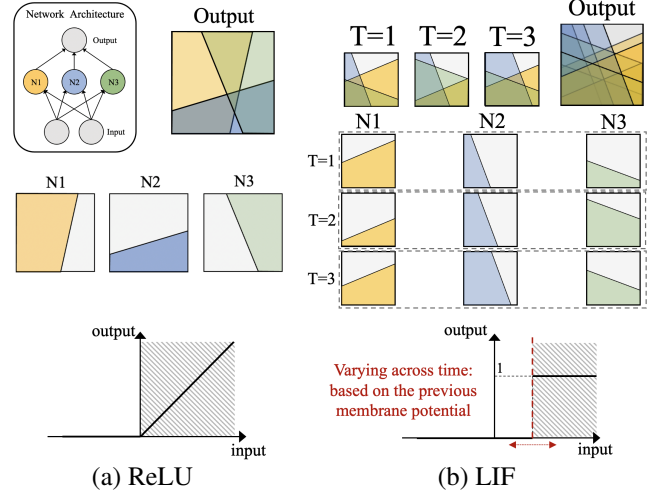


Figure 2. Illustration of the concept of linear regions from ReLU and LIF neurons. Each ReLU (or LIF neuron) divides the two-dimension input space into active and inactive regions. Different from a ReLU neuron, the transfer function of a LIF neuron is changing across time based on the membrane potential from the previous timestep.

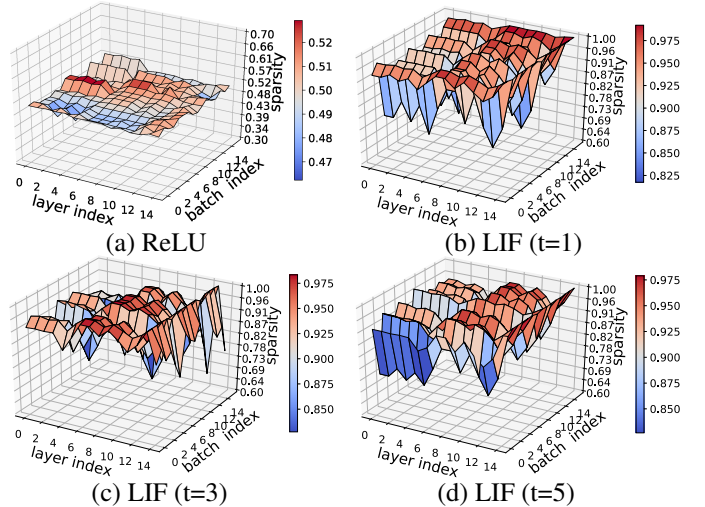


Figure 3. Sparsity variance of activation pattern across different layers and different samples in one mini-batch. LIF neuron shows higher variance of activation pattern compared to ReLU neuron.

activation across all timesteps.

**Observation on the sparsity of activation pattern.** A ReLU neuron provides a binary activation pattern with about 50% sparsity from Gaussian/Uniform weight initialization [34], which is similar across all data samples. On the other hand, a LIF neuron shows a large variation of sparsity across different data samples since the activation pattern is based on the previous membrane potential which is different in each sample. In Fig. 3, we visualize the sparsity of binary activation pattern with 16 mini-batch samples. The results demonstrate that LIF neuron causes a large sparsity variation across different samples in different timesteps.



**A problem due to large sparsity variation.** This large sparsity variation induces different scale of HD. To explain this, we model the output of a LIF neuron at a given timestep for mini-batch data samples  $i$  as Bernoulli distribution where the probability of observing 1 is  $1 - r_i^l$ :

$$o_i^l \sim \text{Bern}(1 - r_i^l). \quad (6)$$

Here,  $r_i^l$  is sparsity of binary activation pattern at layer  $l$ . Then, the probability of output difference between two neurons from mini-batch samples  $i, j$  can be represented as:

$$\Pr(|o_i^l - o_j^l| = 1) = \text{Bern}(r_i^l(1 - r_j^l) + (1 - r_i^l)r_j^l). \quad (7)$$

Therefore, the expectation of HD between two mini-batch samples  $i, j$ :

$$\mathbb{E}[d_H^l(\mathbf{c}_i, \mathbf{c}_j)] = N_A^l [r_i^l(1 - r_j^l) + (1 - r_i^l)r_j^l], \quad (8)$$

where,  $N_A^l$  denotes the number of neurons at layer  $l$ . Note, all quantities in Eq. 6, Eq. 7, and Eq. 8 are evaluated per timestep.

As we can observe in Eq. 8, the expectation of HD is the function of sparsity  $r_i^l$  and  $r_j^l$ . Therefore, HD will provide an inaccurate distance measure for SNN where sparsity  $r^l$  has a large variation across data samples (Fig. 3). For example, HD is likely to be small if two activations are in extreme cases, highly-sparse ( $r \rightarrow 1$ ) or highly-dense ( $r \rightarrow 0$ ). On the other hand, HD is likely to be high if two activations are in a moderate range ( $r \approx 0.5$ ). Thus, based on the sparsity of two activations, HD has a different contribution to the final score  $s$  (Eq. 5); the ideal case is when all HD have the same contribution.

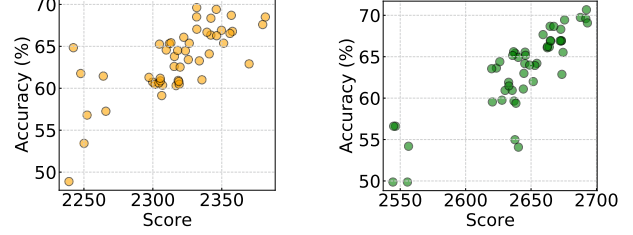
**The proposed solution.** To address this problem, we propose Sparsity-Aware Hamming Distance (SAHD) where Hamming Distance is normalized based on the sparsity of two binary activation patterns. This can be simply done by normalizing the expectation of HD value to a constant  $\alpha$ :

$$d_{SAH}(\mathbf{c}_i, \mathbf{c}_j) = \frac{\alpha}{r_i^l(1 - r_j^l) + (1 - r_i^l)r_j^l} d_H(\mathbf{c}_i, \mathbf{c}_j). \quad (9)$$

Instead of HD, we use SAHD for computing the kernel matrix (Eq. 4) at each timestep. After that, we sum all kernel matrices to compute the final score using Eq. 5. In Fig. 4, we compare the correlation between architecture score and post-training accuracy for HD and SAHD. The results demonstrate that the proposed SAHD has a higher Kendall's  $\tau$  value which implies it is a more accurate metric for architecture selection.

### 4.3. Searching Forward and Backward Connections

Cell-based approach [55, 66, 70, 75, 86, 109] is widely used in NAS research. These methods usually search for the connection topology as well as the corresponding operation



(a) HD (Kendall's  $\tau$ : 0.519) (b) SAHD (Kendall's  $\tau$ : 0.646)

Figure 4. Accuracy with respect to architecture score. We randomly select 50 architectures from search space. We show Kendall's  $\tau$  correlation score for quantitative comparison.

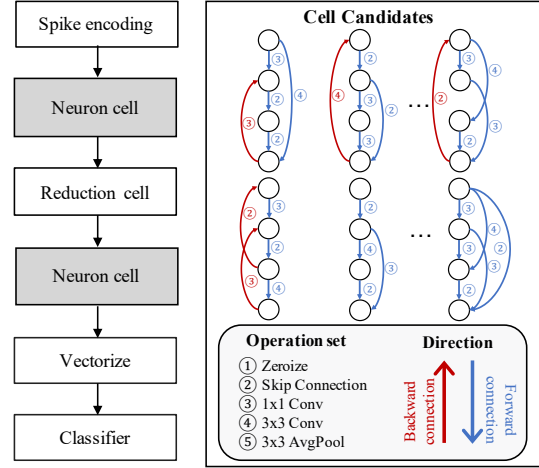


Figure 5. Illustration of cell-based neural architecture search. In search space, we select the optimal spiking neuron cell. Reduction cell downsamples the spatial size of feature map. Interestingly, we found that spiking neuron cell with backward connection achieves better performance than with forward connection only. We do not illustrate Zeroize operation for simplicity.

for each connection. Then, multiple generated cell architectures construct the whole network. In our search algorithm, we also investigate cell-based architectures. Fig. 5 shows the macro skeleton of our SNN architecture. The first block is spike encoding layer which directly convert a float value image into spikes like previous works [85, 101, 106]. The main body of the skeleton consists of two searched neuron cell and one reduction cell. The reduction cell includes one convolution layer and 2-by-2 Average pooling with stride 2. Finally, a linear classifier is used for prediction.

**Cell Search Strategy.** Our cell search space is identical to NAS-Bench-201 [21] (except for backward connections) where each cell includes  $V = 4$  nodes with multiple connections sampled from operation set  $O = \{\text{zeroize}, \text{skip connection}, \text{1-by-1 convolution}, \text{3-by-3 convolution}, \text{3-by-3 average pooling}\}$  (see Fig. 5). Each node contains the sum of all incoming feature maps from edge operation. However, different from [21], we search backward connections in addition to forward connections. In backward operation, we

Table 1. Classification Accuracy (%) on CIFAR10, CIFAR100, and TinyImageNet.

	Dataset	Training Method	Architecture	Timesteps	Accuracy(%)
Sengupta <i>et al.</i> [73]	CIFAR10	ANN-SNN Conversion	VGG16	2500	91.55
Lee <i>et al.</i> [50]	CIFAR10	Surrogate Gradient	VGG9	100	90.45
Rueckauer <i>et al.</i> [72]	CIFAR10	Surrogate Gradient	4Conv, 2Linear	400	90.85
Wu <i>et al.</i> [85]	CIFAR10	Surrogate Gradient	5Conv, 2Linear	12	90.53
Wu <i>et al.</i> [82]	CIFAR10	Tandem Learning	5Conv, 2Linear	8	89.04
Rathi <i>et al.</i> [69]	CIFAR10	Hybrid	VGG9	100	90.50
Han <i>et al.</i> [31]	CIFAR10	ANN-SNN Conversion	VGG16	2048	93.63
Garg <i>et al.</i> [26]	CIFAR10	DCT	VGG9	48	89.94
Kundu <i>et al.</i> [47]	CIFAR10	Hybrid	VGG16	100	91.29
Zheng <i>et al.</i> [106]	CIFAR10	Surrogate Gradient	ResNet19 (doubled #channel)	6	93.16
Deng <i>et al.</i> [18]	CIFAR10	ANN-SNN Conversion	ResNet20	16	92.42
Li <i>et al.</i> [52]	CIFAR10	ANN-SNN Conversion	VGG16	32	93.00
Fang <i>et al.</i> [25]	CIFAR10	Surrogate Gradient	6Conv, 2Linear	8	93.50
Rathi <i>et al.</i> [68]	CIFAR10	Hybrid	ResNet20	10	92.54
Rathi <i>et al.</i> [68]	CIFAR10	Hybrid	VGG16	5	92.70
SNASNet-Fw (ours)	CIFAR10	Surrogate Gradient	Searched Architecture	5	93.12 $\pm$ 0.42
SNASNet-Fw (ours)	CIFAR10	Surrogate Gradient	Searched Architecture	8	93.64 $\pm$ 0.35
SNASNet-Bw (ours)	CIFAR10	Surrogate Gradient	Searched Architecture	5	93.73 $\pm$ 0.32
SNASNet-Bw (ours)	CIFAR10	Surrogate Gradient	Searched Architecture	8	94.12 $\pm$ 0.25
Sengupta <i>et al.</i> [73]	CIFAR100	ANN-SNN Conversion	VGG16	2500	70.90
Lu and Sengupta [57]	CIFAR100	ANN-SNN Conversion	VGG15	62	63.20
Park <i>et al.</i> [64]	CIFAR100	TTFS	VGG15	680	68.80
Rathi <i>et al.</i> [69]	CIFAR100	Hybrid	VGG16	125	67.80
Han <i>et al.</i> [31]	CIFAR100	ANN-SNN Conversion	VGG16	2048	70.90
Garg <i>et al.</i> [26]	CIFAR100	DCT	VGG9	48	68.30
Kundu <i>et al.</i> [47]	CIFAR100	Hybrid	VGG11	120	64.98
Deng <i>et al.</i> [18]	CIFAR100	ANN-SNN Conversion	ResNet20	32	68.40
Li <i>et al.</i> [52]	CIFAR100	ANN-SNN Conversion	ResNet20	16	72.33
Rathi <i>et al.</i> [68]	CIFAR100	Hybrid	ResNet20	5	64.07
Rathi <i>et al.</i> [68]	CIFAR100	Hybrid	VGG16	5	69.67
SNASNet-Fw (ours)	CIFAR100	Surrogate Gradient	Searched Architecture	5	70.06 $\pm$ 0.45
SNASNet-Bw (ours)	CIFAR100	Surrogate Gradient	Searched Architecture	5	73.04 $\pm$ 0.36
Sengupta <i>et al.</i> [73]	TinyImageNet	ANN-SNN Conversion	VGG11	2500	54.20
Kundu <i>et al.</i> [47]	TinyImageNet	Hybrid	VGG16	150	51.92
Garg <i>et al.</i> [26]	TinyImageNet	DCT	VGG13	125	52.43
SNASNet-Fw (ours)	TinyImageNet	Surrogate Gradient	Searched Architecture	5	52.81 $\pm$ 0.56
SNASNet-Bw (ours)	TinyImageNet	Surrogate Gradient	Searched Architecture	5	54.60 $\pm$ 0.48

add transformed node feature of  $l$ -th layer at timestep  $t - 1$  to the node of  $l'$ -th ( $l' < l$ ) layer at timestep  $t$ . The backward connections also have the same operation set search space  $O$  as forward connections. In Fig. 5, we show examples of cell candidates. The forward connections and backward connections can be combined seamlessly. Surprisingly, adding backward connections improves the accuracy of SNNs especially on complex datasets such as CIFAR100 and Tiny-ImageNet. To train the searched SNNs, we use surrogate gradient training [61, 84, 85] (see Supplementary for details).

## 5. Experiments

### 5.1. Implementation Details

**Dataset.** We evaluate our method on CIFAR10 [46], CIFAR100 [46], TinyImageNet [17]. The details of datasets can be found in Supplementary.

**Hyperparameters.** Our implementation is based on PyTorch [65]. We train the networks with standard SGD with momentum 0.9, weight decay 0.0005 and also apply random

crop and horizontal flip to input images. We set batch size for training as 64. The base learning rate is set to 0.2, 0.1, 0.1 for CIFAR10, CIFAR100, TinyImageNet, respectively. We use cosine learning rate scheduling [56]. Here, we set the total number of epochs to 300, 300, 200, for CIFAR10, CIFAR100, TinyImageNet, respectively. We set  $\tau_m$  in Eq. 1 to  $\frac{4}{3}$ . We set  $\alpha$  in Eq. 9 to 0.5 to get similar sparsity scale in LIF neuron as a ReLU neuron. Also, we search 5000 architecture candidates from search space (We observe the accuracy saturates after 5000 samples, shown in Supplementary). We use SpikingJelly [23] package for implementing an LIF neuron.

**Architectures.** Here, we provide details for architectures in Fig. 5. Note, we do not allow two nodes to have both forward and backward connections to ensure training convergence and stability. For the spike encoding layer, we use direct coding [85, 101, 106] where we pass the input image for  $T$  time-steps through the first convolution layer which generates spikes. The first neuron cell has  $C$ -channel input and  $C$ -channel output. Reduction cell consists of  $Conv(C, 2C)$ - $BN(2C)$ - $LIF$  followed by  $AvgPool(2)$ . The second neu-

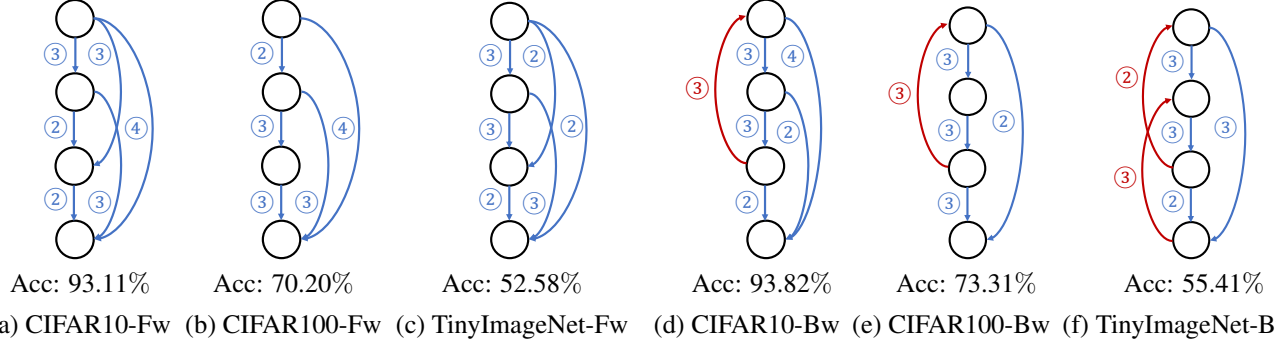


Figure 6. Searched architecture examples (forward and backward configuration) for three benchmarks. Blue and red arrows denote forward connection and backward connection, respectively. The number on each arrow represents operations introduced in Fig. 5.

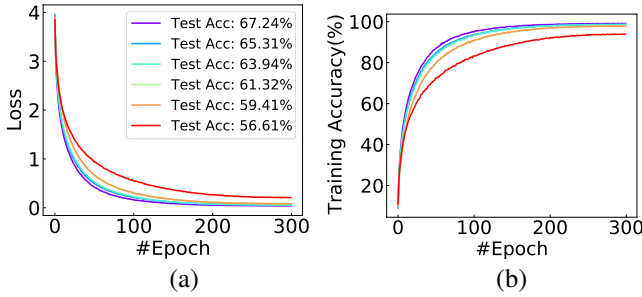


Figure 7. Comparison of the (a) training loss and (b) training accuracy across different searched architectures on CIFAR100.

ron cell has  $2C$ -channel input and  $2C$ -channel output. Note, the structures of the first neuron cell and second neuron cell are identical. We set  $C$  to 256, 128, 128 for CIFAR10, CIFAR100, TinyImageNet, respectively. For vectorize block, we first apply  $\text{AvgPool}(2)$  to the input feature and vectorize the output. Finally, the classifier consists of  $\text{Dropout}(0.5)$ - $\text{FC}(1024)$ - $\text{Voting layer}$ , where a voting layer is used to improve the robustness of classification [25].

## 5.2. Performance Comparison

Table 1 shows the performance comparison between our SNASNet founded by the proposed NAS algorithm and previous SNN models on three benchmarks. As our NAS approach has randomness, we run the same configuration 5 times and report the mean and standard deviation. In the table, “SNASNet-Fw” refers to our searched model with only forward connections and “SNASNet-Bw” denotes our searched model with both forward and backward connections. SNASNet-Fw achieves comparable performance with the previous works with extremely small timesteps. For example, our searched model achieves 70.06% with timestep 5 on CIFAR100, which is similar to the VGG16 model performance from Rathi *et al.* [68]. Note that, for CIFAR10, which is a relatively simple dataset, a few methods yield marginally better performance than SNASNet-Fw. Interestingly, compared to SNASNet-Fw, SNASNet-Bw improves the performance by 0.61%, 2.98%, and 1.79%, for CIFAR10,

CIFAR100, and TinyImageNet, respectively. We note that SNASNet-Bw yields SOTA results across all datasets with only 5/8 timesteps. The results support our assertion that the representation power of SNNs can be enhanced by passing information through backward connections where temporal information is further exploited. We also illustrate the example of searched architecture cell found by our proposed NAS algorithm for each dataset in Fig. 6. Recently, Shu *et al.* [75] show that fast convergence ANN architectures bring smooth loss landscape and accurate gradient information, resulting in high test accuracy. We also found that our searched SNN architectures achieve fast convergence with high test accuracy, as shown in Fig. 7. By using this early stage information, there is a possibility of applying an evolutionary algorithm [70] to SNN searching in future works.

## 5.3. Observations from Searched Cells

In this section, we provide several observations obtained from our searching algorithm. To this end, in Fig. 8, we ran 100 random searches on CIFAR100 and provide averaged accuracy with respect to the number of forward connections (Fw), backward connections (Bw), skip connections, Conv  $3 \times 3$ , Conv  $1 \times 1$ , Average pooling. The key observations are as follows. For **SNASNet with only forward connections**, (1) a deeper and wider cell improves performance, which implies that scaling up SNN is important (1st row in Fig. 8). (2) Convolutional layers are important for getting higher performance. On the other hand, average pooling is not preferred for SNNs. For **SNASNet with both forward and backward connections**, (1) The trend of forward connections (shown in 2nd row) also prefer convolutional layers, which is similar to that of 1st row in Fig. 8. (2) As shown in the 3rd row, a small number of backward connections are preferred. Also, the type of connections does not affect the accuracy except for skip connection. (3) More than 2 backward skip connections degrade accuracy significantly. This implies that feedback without transformation (*e.g.* convolutional or pooling operation) deteriorates representation of SNNs. (4) Considering both forward and backward con-

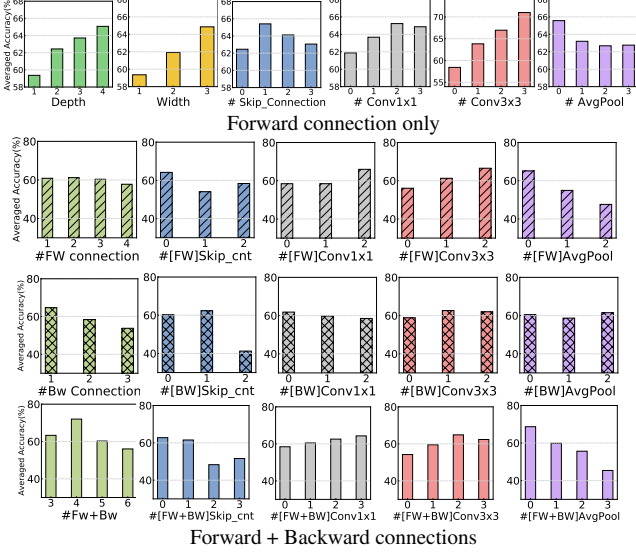


Figure 8. [1st row] Accuracy vs. cell attributes. Searching Fw connection only. [2nd ~ 4th rows] Searching both Fw and Bw connections. 2nd, 3rd, 4th rows show statistics of Fw, Bw, Fw+Bw, respectively.

Table 2. Combining Fang *et al.* [25] with SNASNet-Bw on CIFAR10.

Method	Timestep	Accuracy
Fang <i>et al.</i> [25]	8	93.50
SNASNet-Bw	5	93.73 $\pm$ 0.32
SNASNet-Bw + Fang <i>et al.</i> [25]	5	93.92 $\pm$ 0.41

nections (4th row), the total number of connections should be carefully designed. We conjecture that performance drop with a large number of connections is due to the excessive number of spikes. That is, all neurons are spiking for all inputs that can lead to loss in discrimination.

#### 5.4. Complementary Objective with Prior Works

Existing state-of-the-art SNN works have proposed advanced techniques such as, normalization [106] and learning neuronal dynamics [25] among others, to achieve high-performing SNNs. The motivation of our work is to find high-performing SNNs through architecture search, which is a new line of research compared to previous works focusing on advanced training techniques. Thus, our work is complementary to previous methods. To demonstrate this, we combine our searched architecture with Fang *et al.* [25] where a trainable membrane time constant has been proposed and yields the best accuracy from previous works. In Table 2, our searched architecture is seamlessly combined with such training technique, resulting in further accuracy improvement with even lower timestep requirement. This corroborates our assertion that both careful architectural design and advanced training techniques are important for

Table 3. Transferability study of founded architectures.  $\Delta\text{Acc}$  denotes the performance change compared to the best performed architecture in Fig. 6.

Celltype	Searching dataset	Train/test dataset	Accuracy (%)	$\Delta\text{Acc}$ (%)
Forward	CIFAR10	CIFAR100	69.58	-0.62
Forward	CIFAR10	TinyImageNet	51.96	-0.62
Forward	CIFAR100	CIFAR10	92.02	-0.14
Forward	CIFAR100	TinyImageNet	52.28	-0.30
Forward	TinyImageNet	CIFAR10	92.36	+0.20
Forward	TinyImageNet	CIFAR100	70.31	+0.11
Backward	CIFAR10	CIFAR100	73.09	-0.22
Backward	CIFAR10	TinyImageNet	54.50	-0.91
Backward	CIFAR100	CIFAR10	92.93	-0.11
Backward	CIFAR100	TinyImageNet	56.00	+0.59
Backward	TinyImageNet	CIFAR10	92.84	-0.20
Backward	TinyImageNet	CIFAR100	73.14	-0.17

Table 4. Performance comparison between Hamming Distance (HD) and Sparsity-Aware Hamming Distance (SAHD).

Architecture	HD	SAHD
SNASNet-Fw	64.16 $\pm$ 2.02	70.06 $\pm$ 0.45
SNASNet-Bw	66.80 $\pm$ 1.73	73.04 $\pm$ 0.36

improving SNNs.

#### 5.5. Transferability of Searched Architecture

We conduct transferability analysis on searched SNN architecture in order to check the dependency of our searching method on the dataset. We search the optimal architecture on dataset  $A$  and train/test the searched architecture on dataset  $B$  ( $A \neq B$ ). In Table 3, for both forward and backward configurations, we use the best performed architecture (Fig. 6) for all experiments. The results show that the searched SNASNets are surprisingly transferable across diverse datasets, which opens up promising advantage of eliminating searching time for huge and complex datasets.

#### 5.6. Analysis on Distance Metric

We evaluate the performance of SNASNet according to the distance metric: Hamming Distance (HD) vs. Sparsity-Aware Hamming Distance (SAHD). In Table 4, we report the performance of SNASNet-Fw and SNASNet-Bw on CIFAR100. Here, we also run the search algorithm 5 times and report the mean and standard deviation. The results demonstrate that SAHD reveals better architecture with less standard deviation in terms of test accuracy for both SNASNet-Fw and SNASNet-Bw architectures.

#### 5.7. Ablation on Number of Search Samples

In Fig. 9(a), we report the accuracy with respect to the number of search samples used in our searching algorithm. The backward connection configuration (marked as red) shows higher variation as well as higher performance increase compared to that of the forward connection setting. This is because searching backward connections has larger search space than searching forward connections only. We



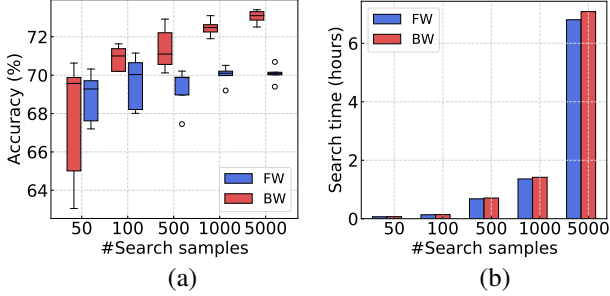


Figure 9. (a) Accuracy and (b) computational time with respect to number of search samples. For accuracy experiments, we run the same settings 5 times. For both experiments, we use CIFAR100 dataset.

Table 5. Ablation studies on timesteps.

Architecture	T=5	T=10	T=15	T=20
SNASNet-Fw	70.06	70.08	70.56	70.52
SNASNet-Bw	73.04	73.46	73.49	74.24

also measure the computational time for searching in Fig. 9(b). We conduct the experiments on NVIDIA RTX 2080ti GPU and Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz processor. The results show that searching backward connection require slightly longer time than searching forward connection. Both forward and backward configurations finish their search process in 8 hours.

### 5.8. Analysis on Timesteps

In Table 5, we report the performance on CIFAR100 with respect to the number of timesteps used in SNNs. Both SNASNet-Fw and SNASNet-Bw achieve performance gain with more number of timesteps. SNASNet-Fw and SNASNet-Bw with 20 timesteps has improved accuracy by 0.48% and 1.2% compared to 5 timesteps, respectively. Interestingly, the performance gain from SNASNet-Bw is larger than SNASNet-Fw. The results suggest that adding backward connections to SNNs effectively leverages the temporal information for improved learning, and thus supports the advantage of backward connections in SNNs.

## 6. Conclusion

For the first time, we search better SNN architecture using the temporal activation pattern of initialized network. Our search space considers backward search connections in addition to forward connections, which brings benefit of using temporal information. So far, SNNs deploy existing ANN models or hand-crafted models designed by a human expert. By achieving better performance than the previous works, we demonstrate that a new type of architecture is more suitable for SNNs where spikes convey information through multiple timesteps. We hope our work fosters future work on searching better SNN-friendly architecture. Finally, we highlight some limitations and possible future directions. (1) Our NAS algorithm uses direct coding, which ensures shorter

timesteps (e.g., 5). However, the proposed method may not be practical or feasible for other coding schemes (such as rate, temporal or phase coding) with  $> 100$  timesteps. Therefore, a faster and more effective NAS SNN search algorithm should be explored for other encoding schemes. (2) Also, our current search space does not search for suitable reduction layer and classifier layer. Defining a proper search space for those blocks can bring better performance as well as efficiency.

## Acknowledgment

The research was funded in part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, the National Science Foundation (Grant#1947826), and Amazon Research Award.

## References

- [1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021.
- [2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.
- [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] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559. PMLR, 2018.
- [5] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24):10464–10472, 1998.
- [6] Tim VP Bliss and Graham L Collingridge. A synaptic model of memory: long-term potentiation in the hippocampus. *Nature*, 361(6407):31–39, 1993.
- [7] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017.
- [8] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.
- [9] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.
- [10] Boyu Chen, Peixia Li, Baopu Li, Chen Lin, Chuming Li, Ming Sun, Junjie Yan, and Wanli Ouyang. Bn-nas: Neural architecture search with batch normalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 307–316, 2021.

- [11] Boyu Chen, Peixia Li, Chuming Li, Baopu Li, Lei Bai, Chen Lin, Ming Sun, Junjie Yan, and Wanli Ouyang. Glit: Neural architecture search for global and local image transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–21, 2021.
- [12] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*, 2021.
- [13] 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*, 32:6642–6652, 2019.
- [14] Dennis V Christensen, Regina Dittmann, Bernabé Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, et al. 2021 roadmap on neuromorphic computing and engineering. *arXiv preprint arXiv:2105.05956*, 2021.
- [15] Iulia M Comsa, Thomas Fischbacher, Krzysztof Potempa, Andrea Gesmundo, Luca Versari, and Jyrki Alakuijala. Temporal coding in spiking neural networks with alpha synaptic function. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8529–8533. IEEE, 2020.
- [16] Vyacheslav Demin and Dmitry Nekhaev. Recurrent spiking neural network learning based on a competitive maximization of neuronal activity. *Frontiers in neuroinformatics*, 12:79, 2018.
- [17] 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.
- [18] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.
- [19] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.
- [20] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.
- [21] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.
- [22] Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5251–5260, 2021.
- [23] Wei Fang, Yanqi Chen, Jianhao Ding, Ding Chen, Zhaoifei Yu, Huihui Zhou, Yonghong Tian, and other contributors. Spikingjelly. <https://github.com/fangwei123456/spikingjelly>, 2020.
- [24] Wei Fang, Zhaoifei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *arXiv preprint arXiv:2102.04159*, 2021.
- [25] Wei Fang, Zhaoifei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2661–2671, 2021.
- [26] Isha Garg, Sayeed Shafayet Chowdhury, and Kaushik Roy. Dct-snn: Using dct to distribute spatial information over time for low-latency spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4671–4680, 2021.
- [27] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3224–3234, 2019.
- [28] Pengjie Gu, Rong Xiao, Gang Pan, and Huajin Tang. Stca: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks. In *IJCAI*, pages 1366–1372, 2019.
- [29] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020.
- [30] Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 388–404. Springer, 2020.
- [31] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13558–13567, 2020.
- [32] Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR, 2019.
- [33] Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. 2019.
- [34] 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.
- [35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.
- [36] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.
- [37] Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining. In *Pro-*

- ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12084–12092, 2020.
- [38] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
  - [39] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.
  - [40] Shuncheng Jia, Tielin Zhang, Xiang Cheng, Hongxing Liu, and Bo Xu. Neuronal-plasticity and reward-propagation improved recurrent spiking neural networks. *Frontiers in Neuroscience*, 15:205, 2021.
  - [41] Xin Jin, Alexander Rast, Francesco Galluppi, Sergio Davies, and Steve Furber. Implementing spike-timing-dependent plasticity on spinnaker neuromorphic hardware. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.
  - [42] Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *arXiv preprint arXiv:2010.01729*, 2020.
  - [43] Youngeun Kim and Priyadarshini Panda. Optimizing deeper spiking neural networks for dynamic vision sensing. *Neural Networks*, 2021.
  - [44] Youngeun Kim and Priyadarshini Panda. Visual explanations from spiking neural networks using interspike intervals. *Sci Rep* 11, 19037 (2021). <https://doi.org/10.1038/s41598-021-98448-0>, 2021.
  - [45] Youngeun Kim, Yeshwanth Venkatesha, and Priyadarshini Panda. Privatesnn: Fully privacy-preserving spiking neural networks. *arXiv preprint arXiv:2104.03414*, 2021.
  - [46] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
  - [47] Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A Beerel. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3953–3962, 2021.
  - [48] Souvik Kundu, Massoud Pedram, and Peter A Beerel. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5209–5218, 2021.
  - [49] Eimantas Ledinauskas, Julius Ruseckas, Alfonsas Juršėnas, and Giedrius Buračas. Training deep spiking neural networks. *arXiv preprint arXiv:2006.04436*, 2020.
  - [50] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 14, 2020.
  - [51] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.
  - [52] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. *arXiv preprint arXiv:2106.06984*, 2021.
  - [53] Ling Liang, Zheng Qu, Zhaodong Chen, Fengbin Tu, Yujie Wu, Lei Deng, Guoqi Li, Peng Li, and Yuan Xie. H2learn: High-efficiency learning accelerator for high-accuracy spiking neural networks. *arXiv preprint arXiv:2107.11746*, 2021.
  - [54] 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.
  - [55] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
  - [56] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.
  - [57] Sen Lu and Abhronil Sengupta. Exploring the connection between binary and spiking neural networks. *Frontiers in Neuroscience*, 14:535, 2020.
  - [58] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021.
  - [59] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *arXiv preprint arXiv:1402.1869*, 2014.
  - [60] Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2017.
  - [61] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36:61–63, 2019.
  - [62] Priyadarshini Panda, Sai Aparna Aketi, and Kaushik Roy. Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization. *Frontiers in Neuroscience*, 14, 2020.
  - [63] Priyadarshini Panda and Kaushik Roy. Learning to generate sequences with combination of hebbian and non-hebbian plasticity in recurrent spiking neural networks. *Frontiers in neuroscience*, 11:693, 2017.
  - [64] Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: Deep spiking neural networks with time-to-first-spike coding. *arXiv preprint arXiv:2003.11741*, 2020.
  - [65] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.
  - [66] 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.
  - [67] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR, 2017.

- [68] Nitin Rathi and Kaushik Roy. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
- [69] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent back-propagation. *arXiv preprint arXiv:2005.01807*, 2020.
- [70] 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.
- [71] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.
- [72] 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.
- [73] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.
- [74] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *arXiv preprint arXiv:1810.08646*, 2018.
- [75] Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. *arXiv preprint arXiv:1909.09569*, 2019.
- [76] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020.
- [77] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.
- [78] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.
- [79] Yeshwanth Venkatesha, Youngeun Kim, Leandros Tassiulas, and Priyadarshini Panda. Federated learning with spiking neural networks. *arXiv preprint arXiv:2106.06579*, 2021.
- [80] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.
- [81] Hao Wu, Yueyi Zhang, Wenming Weng, Yongting Zhang, Zhiwei Xiong, Zheng-Jun Zha, Xiaoyan Sun, and Feng Wu. Training spiking neural networks with accumulated spiking flow. *ijo*, 1(1), 2021.
- [82] Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *arXiv e-prints*, pages arXiv–1907, 2019.
- [83] Jibin Wu, Chenglin Xu, Daquan Zhou, Haizhou Li, and Kay Chen Tan. Progressive tandem learning for pattern recognition with deep spiking neural networks. *arXiv preprint arXiv:2007.01204*, 2020.
- [84] 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.
- [85] 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.
- [86] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.
- [87] Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. On the number of linear regions of convolutional neural networks. In *International Conference on Machine Learning*, pages 10514–10523. PMLR, 2020.
- [88] Jingjing Xu, Liang Zhao, Junyang Lin, Rundong Gao, Xu Sun, and Hongxia Yang. Knas: green neural architecture search. In *International Conference on Machine Learning*, pages 11613–11625. PMLR, 2021.
- [89] Lumin Xu, Yingda Guan, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang. Vipnas: Efficient video pose estimation via neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16072–16081, 2021.
- [90] Qi Xu, Jianxin Peng, Jiangrong Shen, Huajin Tang, and Gang Pan. Deep covdensenn: A hierarchical event-driven dynamic framework with spiking neurons in noisy environment. *Neural Networks*, 121:512–519, 2020.
- [91] Zhicheng Yan, Xiaoliang Dai, Peizhao Zhang, Yuandong Tian, Bichen Wu, and Matt Feiszli. Fp-nas: Fast probabilistic neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15139–15148, 2021.
- [92] Tien-Ju Yang, Yi-Lun Liao, and Vivienne Sze. Netadaptv2: Efficient neural architecture search with fast super-network training and architecture optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2402–2411, 2021.
- [93] Yibo Yang, Shan You, Hongyang Li, Fei Wang, Chen Qian, and Zhouchen Lin. Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6667–6676, 2021.
- [94] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Jianyuan Guo, Wei Zhang, Chao Xu, Chunjing Xu, Dacheng Tao, and Chang Xu. Hournas: Extremely fast neural architecture search through an hourglass lens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10896–10906, 2021.
- [95] Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhaoxu Yang, and Guoqi Li. Temporal-wise



- attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10221–10230, 2021.
- [96] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019.
- [97] Amirreza Yousefzadeh, Evangelos Stromatias, Miguel Soto, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. On practical issues for stochastic stdp hardware with 1-bit synaptic weights. *Frontiers in neuroscience*, 12:665, 2018.
- [98] Davide Zambrano, Roeland Nusselder, H Steven Scholte, and Sander M Bohtë. Sparse computation in adaptive spiking neural networks. *Frontiers in neuroscience*, 12:987, 2019.
- [99] Dan Zeng, Yuhang Huang, Qian Bao, Junjie Zhang, Chi Su, and Wu Liu. Neural architecture search for joint human parsing and pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11385–11394, 2021.
- [100] Wenrui Zhang and Peng Li. Spike-train level backpropagation for training deep recurrent spiking neural networks. *arXiv preprint arXiv:1908.06378*, 2019.
- [101] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *arXiv preprint arXiv:2002.10085*, 2020.
- [102] Xuanyang Zhang, Pengfei Hou, Xiangyu Zhang, and Jian Sun. Neural architecture search with random labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10907–10916, 2021.
- [103] Xinbang Zhang, Zehao Huang, Naiyan Wang, Shiming Xiang, and Chunhong Pan. You only search once: Single shot neural architecture search via direct sparse optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):2891–2904, 2020.
- [104] Xiong Zhang, Hongmin Xu, Hong Mo, Jianchao Tan, Cheng Yang, Lei Wang, and Wenqi Ren. Dcnas: Densely connected neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13956–13967, 2021.
- [105] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. In *International Conference on Machine Learning*, pages 12707–12718. PMLR, 2021.
- [106] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. *arXiv preprint arXiv:2011.05280*, 2020.
- [107] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2423–2432, 2018.
- [108] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.
- [109] 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.