Exploring Lottery Ticket Hypothesis in Spiking Neural Networks

Youngeun Kim⬤, Yuhang Li⬤, Hyoungseob Park⬤, Yeshwanth Venkatesha,
Ruokai Yin⬤, and Priyadarshini Panda⬤

Department of Electrical Engineering
Yale University
New Haven, CT, USA
{youngeun.kim, yuhang.li, hyoungseob.park, yeshwanth.venkatesha,
ruokai.yin, priya.panda}@yale.edu

Abstract. Spiking Neural Networks (SNNs) have recently emerged as a new generation of low-power deep neural networks, which is suitable to be implemented on low-power mobile/edge devices. As such devices have limited memory storage, neural pruning on SNNs has been widely explored in recent years. Most existing SNN pruning works focus on shallow SNNs (2∼6 layers), however, deeper SNNs (≥16 layers) are proposed by state-of-the-art SNN works, which is difficult to be compatible with the current SNN pruning work. To scale up a pruning technique towards deep SNNs, we investigate Lottery Ticket Hypothesis (LTH) which states that dense networks contain smaller subnetworks (*i.e.*, winning tickets) that achieve comparable performance to the dense networks. Our studies on LTH reveal that the winning tickets consistently exist in deep SNNs across various datasets and architectures, providing up to 97% sparsity without huge performance degradation. However, the iterative searching process of LTH brings a huge training computational cost when combined with the multiple timesteps of SNNs. To alleviate such heavy searching cost, we propose Early-Time (ET) ticket where we find the important weight connectivity from a smaller number of timesteps. The proposed ET ticket can be seamlessly combined with a common pruning techniques for finding winning tickets, such as Iterative Magnitude Pruning (IMP) and Early-Bird (EB) tickets. Our experiment results show that the proposed ET ticket reduces search time by up to 38% compared to IMP or EB methods. Code is available at Github.

Keywords: Spiking Neural Networks, Neural Network Pruning, Lottery Ticket Hypothesis, Neuromorphic Computing

1 Introduction

Spiking Neural Networks (SNNs) [63,11,74,75,37,20,76,34] have gained significant attention as promising low-power alternative to Artificial Neural Networks (ANNs). Inspired by the biological neuron, SNNs process visual information through discrete spikes over multiple timesteps. This event-driven behavior of
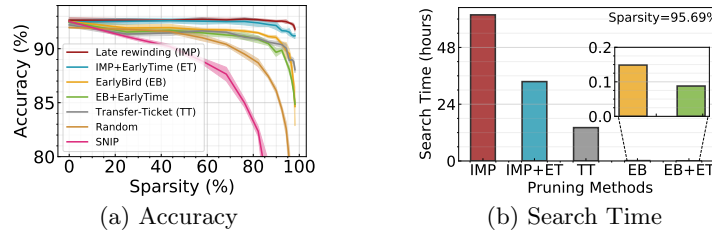
(a) Accuracy   (b) Search Time

Fig. 1. Accuracy and Search time comparison of various pruning methods on SNNs including Iterative Magnitude Pruning (IMP) [21], Early-Bird (EB) ticket [80], Early-Time (ET) ticket (ours), Transferred winning ticket from ANN (TT), Random pruning, and SNIP [41]. We use VGG16 on the CIFAR10 dataset and show mean/standard deviation from 5 random runs.

SNNs brings huge energy-efficiency, therefore they are suitable to be implemented on low-power neuromorphic chips [1,13,23,57] which compute spikes in an asynchronous manner. However, as such devices have limited memory storage, neural pruning can be one of the essential techniques by reducing memory usage for weight parameters, thus promoting the practical deployment.

Accordingly, researchers have made certain progress on the pruning technique for SNNs. Neftci *et al.* [55] and Rathi *et al.* [60] prune weight connections of SNNs using a predefined threshold value. Guo *et al.* [25] propose an unsupervised online adaptive weight pruning algorithm that dynamically removes non-critical weights over time. Moreover, Shi *et al.* [65] present a soft-pruning method where both weight connections and a pruning mask are trained during training. Recently, Deng *et al.* [14] adapt ADMM optimization tool with sparsity regularization to compress SNNs. Chen *et al.* [9] propose a gradient-based rewiring method for pruning, where weight values and connections are jointly optimized. However, although existing SNN pruning works significantly increase weight sparsity, they focus on a shallow architecture such as 2-layer MLP [55,60,25] or 6∼7 convolutional layers [14,9]. Such pruning techniques are difficult to scale up to the recent state-of-the-art deep SNN architectures where the number of parameters and network depth is scaled up [43,44,20,61,83].

In this paper, we explore sparse deep SNNs based on the recently proposed Lottery Ticket Hypothesis (LTH) [21]. They assert that an over-parameterized neural network contains sparse sub-networks that achieve a similar or even better accuracy than the original dense networks. The discovered sub-networks and their corresponding initialization parameters are referred as *winning tickets*. Based on LTH, a line of works successfully have shown the existence of winning tickets across various tasks such as standard recognition task [80,22,24], reinforcement learning [69,81], natural language processing [4,7,53], and generative model [31]. Along the same line, our primary research objective is to investigate the existence of winning tickets in deep SNNs which has a different type of neuronal dynamics from the common ANNs.

Furthermore, applying LTH on SNNs poses a practical challenge. In general, finding winning tickets requires **Iterative Magnitude Pruning (IMP)** where a network becomes sparse by repetitive initialization-training-pruning operations [21]. Such iterative training process goes much slower with SNNs where multiple feedforward steps (*i.e.*, timesteps) are required. To make LTH with SNNs more practical, we explore several techniques for reducing search costs. We first investigate **Early-Bird (EB) ticket** phenomenon [80] that states the sub-networks can be discovered in the early-training phase. We find that SNNs contain EB tickets across various architectures and datasets. Moreover, SNNs convey information through multiple timesteps, which provides a new dimension for computational cost reduction. Focusing on such temporal property, we propose **Early-Time (ET) ticket** phenomenon: winning tickets can be drawn from the network trained from a smaller number of timesteps. Thus, during the search process, SNNs use a smaller number of timesteps, which can significantly reduce search costs. As ET ticket is a temporal-crafted method, our proposed ET ticket can be combined with IMP [21] and EB tickets [80]. Furthermore, we also explore whether **ANN winning ticket can be transferred to SNN** since search cost at ANN is much cheaper than SNN. Finally, we examine **pruning at initialization** method on SNNs, *i.e.*, SNIP [41], which finds the winning tickets from backward gradients at initialization. In Fig. 1, we compare the accuracy and search time of the above-mentioned pruning methods.

In summary, we explore LTH for SNNs by conducting extensive experiments on two representative deep architectures, *i.e.*, VGG16 [67] and ResNet19 [28], on four public datasets including SVHN [56], Fashion-MNIST [77], CIFAR10 [36] and CIFAR100 [36]. Our key observations are as follows:

- We confirm that Lottery Ticket Hypothesis is valid for SNNs.
- We found that IMP [21] discovers winning tickets with up to 97% sparsity. However, IMP requires over 50 hours on GPU to find sparse ($\geq$ 95%) SNNs.
- EB tickets [80] discover sparse SNNs in 1 hour on GPUs, which reduces search cost significantly compared to IMP. Unfortunately, they fail to detect winning tickets over 90% sparsity.
- Applying ET tickets to both IMP and EB significantly reduces search time by up to 41% while showing $\leq$ 1% accuracy drop at less than 95% sparsity.
- Winning ticket obtained from ANN can be transferable at less than 90% sparsity. However, huge accuracy drop is incurred at high sparsity levels ($\geq$ 95%), especially for complex datasets such as CIFAR100.
- Pruning at initialization method [41] fails to discover winning tickets in SNNs owing to the non-differentiability of spiking neurons.

2   Related Work

2.1   Spiking Neural Networks

Spiking Neural Networks (SNNs) transfer binary and asynchronous information through networks in a low-power manner [63,11,12,52,64,43,68,78]. The major

difference of SNNs from standard ANNs is using a Leak-Integrate-and-Fire (LIF) neuron [30] as a non-linear activation. The LIF neuron accumulates incoming spikes in membrane potential and generates an output spike when the neuron has a higher membrane potential than a firing threshold. Such integrate-and-fire behavior brings a non-differentiable input-output transfer function where standard backpropagation is difficult to be applied [54]. Recent SNN works circumvent non-differentiable backpropagation problem by defining a surrogate function for LIF neurons when calculating backward gradients [40,39,54,66,74,72,45,32,73]. Our work is also based on a gradient backpropagation method with a surrogate function (details are provided in Supplementary A). The gradient backpropagation methods enable SNNs to have deeper architectures. For example, adding Batch Normalization (BN) [29] to SNNs [38,33,83] improves the accuracy with deeper architectures such as VGG16 and ResNet19. Also, Fang *et al.* [20] revisit deep residual connection for SNNs, showing higher performance can be achieved by adding more layers. Although the recent state-of-the-art SNN architecture goes deeper [83,20,15], pruning for such networks has not been explored. We assert that showing the existence of winning tickets in deep SNNs brings a practical advantage to resource-constrained neuromorphic chips and edge devices.

2.2 Lottery Ticket Hypothesis

Pruning has been actively explored in recent decades, which compresses a huge model size of the deep neural networks while maintaining its original performance [27,26,71,47,42]. In the same line of thought, Frankle & Carbin [21] present Lottery Ticket Hypothesis (LTH) which states that an over-parameterized neural network contains sparse sub-networks with similar or even better accuracy than the original dense networks. They search winning tickets by iterative magnitude pruning (IMP). Although IMP methods [84,2,18,5,46] provide higher performance compared to existing pruning methods, such iterative training-pruning-retraining operations require a huge training cost. To address this, a line of work [50,51,16,7] discovers the existence of transferable winning tickets from the source dataset and successfully transfers it to the target dataset, thus eliminating search cost. Further, You *et al.* [80] introduce *early bird ticket* hypothesis where they conjecture winning tickets can be achieved in the early training phase, reducing the cost for training till convergence. Recently, Zhang *et al.* [82] discover winning tickets with a carefully selected subset of training data, called pruning-aware critical set. To completely eliminate training costs, several works [41,70] present searching algorithms from initialized networks, which finds winning tickets without training. Unfortunately, such techniques do not show comparable performance with the original IMP methods, thus mainstream LTH leverages IMP as a pruning scheme [24,82,51]. Based on IMP technique, researchers found the existence of LTH in various applications including visual recognition tasks [24], natural language processing [4,7,53], reinforcement learning [69,81], generative model [31], low-cost neural network ensembling [46], and improving robustness [8]. Although LTH has been actively explored in ANN domain, LTH for SNNs is rarely studied. It is worth mentioning that Martinelli *et*
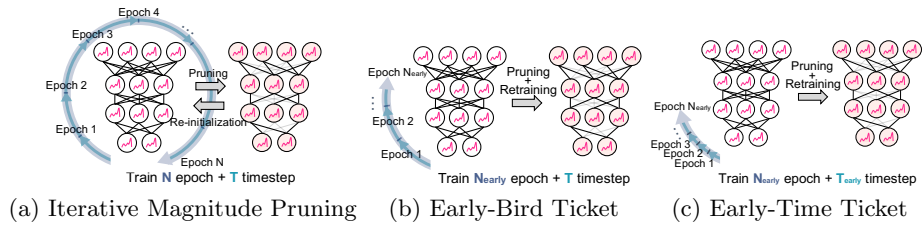
(a) Iterative Magnitude Pruning   (b) Early-Bird Ticket   (c) Early-Time Ticket

Fig. 2. Illustration of the concept of Iterative Magnitude Pruning (IMP), Early-Bird (EB) ticket, and the proposed Early-Time (ET) ticket applied to EB ticket. Our ET ticket reduces search cost for winning tickets by using a smaller number of timesteps during the search process. Note, ET can be applied to both IMP and EB, here we only illustrate ET with EB.

al. [49] apply LTH to two-layer SNN on voice activity detection task. Different from the previous work, our work shows the existence of winning tickets in much deeper networks such as VGG16 and ResNet19, which shows state-of-the-art performance on the image recognition task. We also explore Early-Bird ticket [80], SNIP [41], transferability of winning tickets from ANN, and propose a new concept of winning tickets in the temporal dimension.

3   Drawing Winning Tickets from SNN

In this section, we present the details of pruning methods based on LTH, explored in our experiments. We first introduce the LTH [21] and Early-Bird (EB) Ticket [80]. Then, we propose Early-Time (ET) Tickets where we reduce search cost in the temporal dimension of SNNs. We illustrate the overall search process of each method in Fig. 2.

3.1   Lottery Ticket Hypothesis

In LTH [21], winning tickets are discovered by iterative magnitude pruning (IMP). The whole pruning process of LTH goes through K iterations for the target pruning ratio $p^K\%$. Consider a randomly initialized dense network $f(x;\theta)$ where $\theta \in \mathbb{R}^n$ is the network parameter weights. For the first iteration, initialized network $f(x;\theta)$ is trained till convergence, then mask $m_1 \in \{0,1\}^n$ is generated by removing $p\%$ lowest absolute value weights parameters. Given a pruning mask m_1, we can define subnetworks $f(x;\theta \odot m_1)$ by removing some connections. For the next iteration, we reinitialize the network with $\theta \odot m_1$, and prune p% weights when the network is trained to convergence. This pruning process is repeated for K iterations. In our experiments, we set p and K to 25% and 15, respectively. Also, Frankle *et al.* [22] present *Late Rewinding*, which rewinds the network to the weights at epoch i rather than initialization. This enables IMP to discover the winning ticket with less performance drop in a high

sparsity regime by providing a more stable starting point. We found that *Late Rewinding* shows better performance than the original IMP in deep SNNs (see Supplementary B). Throughout our paper, we apply *Late Rewinding* to IMP for experiments where we rewind the network to epoch 20.

3.2   Early-Bird Tickets

Using IMP for finding lottery ticket incurs huge computational costs. To address this, You *et al.* [80] propose an efficient pruning method, called Early-bird (EB) tickets, where they show winning tickets can be discovered at an early training epoch. Specifically, at searching iteration k, they obtain a mask m_k and measure the mask difference between the current mask m_k and the previous masks within time window q, *i.e.*, $m_{k-1}, m_{k-2}, ..., m_{k-q}$. If the maximum difference is less than hyperparamter τ, they stop training and use m_k as an EB ticket. As searching winning tickets in SNN takes a longer time than ANN, we explore the existence of EB tickets in SNNs. In our experiments, we set q and τ to 5 and 0.02, respectively. Although the original EB tickets use channel pruning, we use unstructured weight pruning for finding EB tickets in order to achieve a similar sparsity level with other pruning methods used in our experiments.

3.3   Early-Time Tickets

Even though EB tickets significantly reduce searching time for winning tickets. For SNNs, one image is passed to a network through multiple timesteps, which provides a new dimension for computational cost reduction. We ask: *can we find the important weight connectivity for the SNN trained with timestep T from the SNN trained with shorter timestep $T' < T$?*

Preliminary Experiments. To answer this question, we conduct experiments on two representative deep architectures (VGG16 and ResNet19) on two datasets (CIFAR10 and CIFAR100). Our experiment protocol is shown in Fig. 3.3 (left panel). We first train the networks with timestep T_{pre} till convergence. After that, we prune $p\%$ of the low-magnitude weights and re-initialize the networks. Finally, we re-train the pruned networks with longer timestep $T_{post} > T_{pre}$, and measure the test accuracy. Thus, this experiment shows the performance of SNNs where the structure is obtained from the lower timestep. In our experiments, we set $T_{pre} = \{2, 3, 4, 5\}$ and $T_{post} = 5$. Surprisingly, the connections founded from $T_{pre} \geq 3$ can bring similar and even better accuracy compared to the unpruned baseline, as shown in Fig. 3.3. Note, in the preliminary experiments, we use a common post-training pruning based on the magnitude of weights [27]. Thus, we can extrapolate the existence of early-time winning tickets to a more sophisticated pruning method such as IMP [21], which generally shows better performance than post-training pruning. We call such a winning ticket as *Early-Time tickets*; a winning ticket drawn with a trained network from a smaller number of timesteps T_{early}, which shows matching performance with a winning ticket from the original timesteps T.
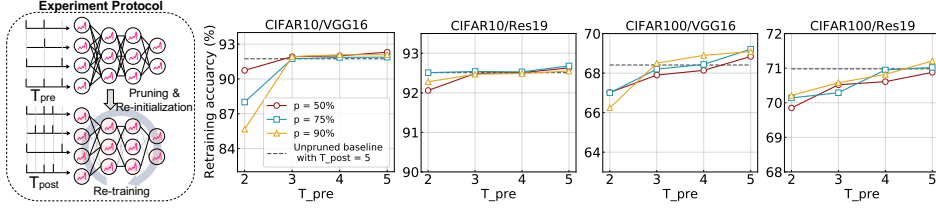
Fig. 3. Preliminary experiments for Early-Time ticket. We conduct experiments on VGG16/ResNet19 on CIFAR10/CIFAR100. We report retraining accuracy ($T_{post} = 5$) with respect to the timestep (T_{pre}) for searching the important connectivity in SNNs.
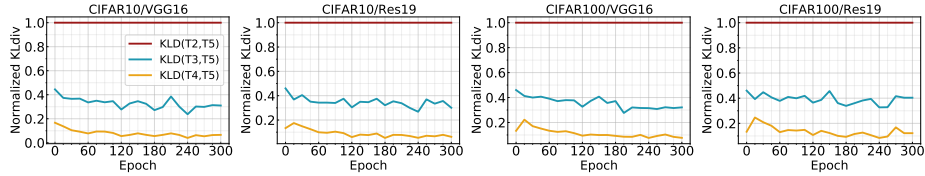

Fig. 4. Kullback–Leibler (KL) divergence between the class prediction distribution from different timesteps. The network is trained with the original timestep $T = 5$. We measure KL divergence between the predicted class probabilities from different timesteps. We use the training set for calculating KL divergence.

Proposed Method.   Then, *how to practically select a timestep for finding Early-Time tickets?* The main idea is to measure the similarity between class predictions between the original timestep T and a smaller number of timesteps, and select a minimal timestep that shows a similar representation with the target timestep. Specifically, let P_T be the class probability from the last layer of networks by accumulating output values across T timesteps [39]. In this case, our search space can be $S = \{2, 3, ..., T-1\}$. Note, timestep 1 is not considered since it cannot use the temporal behavior of LIF neurons. To measure the statistical distance between class predictions P_T and $P_{T'}$, we use Kullback–Leibler (KL) divergence:

$$D_{KL}(P_{T'}||P_T) = \sum_x P_{T'}(x)\ln\frac{P_{T'}(x)}{P_T(x)}. \tag{1}$$

The value of KL divergence goes smaller when the timestep T' is closer to the original timestep T, *i.e.*, $D_{KL}(P_{T-1}||P_T) \leq D_{KL}(P_{T-2}||P_T) \leq ... \leq D_{KL}(P_2||P_T)$. Note, for any $t \in \{1, .., T\}$, we compute P_t by accumulating output layer's activation from 1 to t timesteps. Therefore, due to the accumulation, if the timestep difference between timestep t and t' becomes smaller, the KL divergence $D_{KL}(P_{t'}||P_t)$ becomes lower. After that, we rescale all KL divergence values to [0, 1] by dividing them with $D_{KL}(P_2||P_T)$. In Fig. 3.3, we illustrate normalized KL divergence of VGG16 and ResNet19 on CIFAR10 and CIFAR100,

---

Algorithm 1 Early-Time (ET) ticket

Input: Training data D; Winning ticket searching method $F(\cdot)$ – IMP or EB ticket;
Original timestep T; Threshold λ
Output: Pruned SNN_{pruned}

1: Training SNN with N epochs for stability
2: Memory $= [\,]$
3: **for** $t \leftarrow 2$ to T **do**
4:      $P_t \leftarrow SNN(t, D)$ ▷ Storing class prediction from each timestep
5:      Memory.append(P_t)
6: **end for**
7: $[D_{KL}(P_{T-1}||P_T), ..., D_{KL}(P_2||P_T)] \leftarrow$ Memory   ▷ Computing KL div.
8: **for** $t \leftarrow 2$ to $T - 1$ **do**
9:      $\hat{D}_{KL}(P_t||P_T) \leftarrow \frac{D_{KL}(P_t||P_T)}{D_{KL}(P_2||P_T)}$ ▷ Normalization
10:      **if** $\hat{D}_{KL}(P_t||P_T) < \lambda$ **then**   ▷ Select timestep when KL div. is less than λ
11:          $T_{early} = t$
12:          **break**
13:      **end if**
14: **end for**
15: $WinningTicket \leftarrow F(SNN, T_{early})$ ▷ Finding winning ticket with T_{early}
16: $SNN_{pruned} \leftarrow F(WinningTicket, T)$ ▷ Train with the original timestep T
17: **return** SNN_{pruned}

---

where we found two observations: (1) The KL divergence with $T = 2$ has a relatively higher value than other timesteps. If the difference in the class probability is large (*i.e.*, larger KL divergence), weight connections are likely to be updated in a different direction. This supports our previous observation that important connectivity founded by $T = 2$ shows huge performance drop at $T = 5$ (Fig. 3.3). Therefore, we search T_{early} that has KL divergence less than λ while minimizing the number of timesteps. The λ is a hyperparameter for the determination of T_{early} in the winning ticket search process (Algorithm 1, line 15). A higher λ leads to smaller T_{early} and vice versa (we visualize the change of T_{early} versus different λ values in Fig. 7(b)). For example, if we set $\lambda = 0.6$, timestep 3 is used for finding early-time tickets. In our experiments, we found that a similar value of threshold λ can be applied across various datasets. (2) The normalized KL divergence shows fairly consistent values across training epochs. Thus, we can find the suitable timestep T_{early} for obtaining early-time tickets at the very beginning of the training phase.

The early-time ticket approach can be seamlessly applied to both IMP and Early-bird ticket methods. Algorithm 1 illustrates the overall process of Early-Time ticket. For stability with respect to random initialization, we start to search T_{early} after $N = 2$ epoch of training (Line 1). We show the variation of KL divergence results across training epochs in Supplementary C. We first find the T_{early} from KL Divergence of difference timesteps (Line 2-13). After that, we discover the winning ticket using either IMP or EB ticket with T_{early} (Line 14). Finally, the winning ticket is trained with the original timestep T (Line 15).
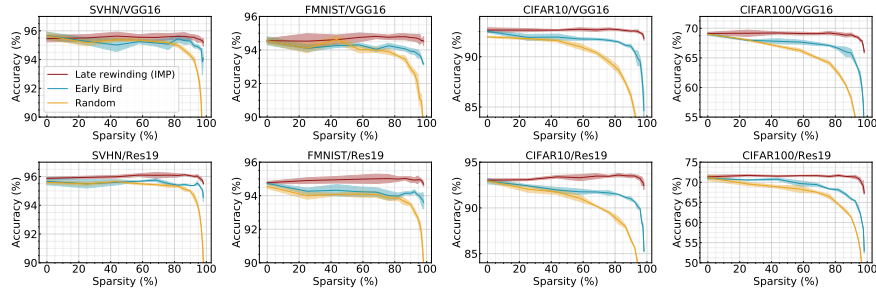
Fig. 5. The accuracy of winning ticket with respect to sparsity level. We report mean and standard deviation from 5 random runs.

Table 1. Effect of the proposed Early-Time ticket. We compare the accuracy and search time of Iterative Magnitude Pruning (IMP), Early-Bird (EB) ticket, Early-Time (ET) ticket on four sparsity levels. We show search speed gain and accuracy change from applying ET.

Setting	Method	Accuracy				Winning Ticket Search Time (hours)			
		p=68.30%	p=89.91%	p=95.69%	p=98.13%	p=68.30%	p=89.91%	p=95.69%	p=98.13%
CIFAR10 VGG16	IMP	92.66	92.54	92.38	91.81	14.97	29.86	40.84	51.99
	IMP + ET	92.49	92.09	91.54	91.10	11.19	22.00	30.11	38.26
	Δ Acc. / Speed Gain	-0.17	-0.45	-0.84	-0.71	×1.34	×1.35	×1.35	×1.35
	EB	91.74	91.05	89.55	84.64	1.96	0.74	0.11	0.09
	EB + ET	91.27	90.66	88.95	84.86	1.44	0.55	0.07	0.06
	Δ Acc. / Speed Gain	-0.47	-0.39	-0.60	+0.22	×1.36	×1.34	×1.18	×1.12
CIFAR10 Res19	IMP	93.47	93.49	93.22	92.43	21.01	42.20	58.91	73.54
	IMP + ET	93.10	92.72	92.68	91.36	13.35	26.62	37.27	46.40
	Δ Acc. / Speed Gain	-0.37	-0.77	-0.54	-1.07	×1.57	×1.59	×1.58	×1.58
	EB	91.00	90.84	89.90	85.22	2.49	0.87	0.24	0.08
	EB + ET	90.83	91.21	89.65	85.45	1.63	0.58	1.70	0.07
	Δ Acc. / Speed Gain	-0.17	+0.37	-0.50	-1.09	×1.52	×1.49	×1.38	×1.16
CIFAR100 VGG16	IMP	69.08	68.90	68.00	66.02	15.02	29.99	41.03	52.05
	IMP + ET	68.27	67.99	66.51	64.41	11.24	22.42	30.53	38.32
	Δ Acc. / Speed Gain	-0.81	-0.91	-1.49	-1.61	×1.33	×1.34	×1.34	×1.36
	EB	67.35	65.82	61.90	52.11	2.27	0.99	0.32	0.06
	EB + ET	67.26	64.18	61.81	52.77	1.66	0.73	0.24	0.05
	Δ Acc. / Speed Gain	-0.09	-1.64	-0.09	+0.66	×1.36	×1.35	×1.31	×1.12
CIFAR100 ResNet19	IMP	71.64	71.38	70.45	67.35	21.21	42.29	59.17	73.52
	IMP + ET	71.06	70.45	69.23	65.49	13.56	27.05	37.88	46.65
	Δ Acc. / Speed Gain	-0.58	-0.93	-1.22	-1.86	×1.56	×1.56	×1.56	×1.57
	EB	69.41	65.87	62.18	52.92	3.08	1.71	0.43	0.09
	EB + ET	68.98	65.76	62.20	51.50	2.00	1.12	0.29	0.07
	Δ Acc. / Speed Gain	-0.43	-0.12	+0.02	-1.42	×1.53	×1.52	×1.45	×1.16

4   Experimental Results

4.1   Implementation Details

We comprehensively evaluate various pruning methods on four public datasets: SVHN [56], Fashion-MNIST [77], CIFAR10 [36] and CIFAR100 [36]. In our work, we focus on pruning deep SNNs, therefore we evaluate two representative architectures; VGG16 [67] and ResNet19 [28]. Our implementation is based on PyTorch [59]. We train the network using SGD optimizer with momentum 0.9 and weight decay 5e-4. Our image augmentation process and loss function follow the previous SNN work [15]. We set the training batch size to 128. The base learning

rate is set to 0.3 for all datasets with cosine learning rate scheduling [48]. Here, we set the total number of epochs to 150, 150, 300, 300 for SVHN, F-MNIST, CIFAR10, CIFAR100, respectively. We set the default timesteps T to 5 across all experiments. Also, we use $\lambda = 0.6$ for finding Early-Time tickets. Experiments were conducted on an RTX 2080Ti GPU with PyTorch implementation. We use SpikingJelly [19] package for implementation.

4.2   Winning Tickets in SNNs

Performance of IMP and EB ticket. In Fig. 5, we show the performance of the winning tickets from IMP and EB. The performance of random pruning is also provided as a reference. Both IMP and EB can successfully find the winning ticket, which shows better performance than random pruning. Especially, IMP finds winning tickets over $\sim 97\%$ sparsity across all configurations. Also, we observe that the winning ticket sparsity is affected by dataset complexity. EB ticket can find winning tickets ($> 95\%$ sparsity) for relatively simple datasets such as SVHN and Fashion-MINST. However, they are limited to discovering the winning ticket having $\leq 95\%$ sparsity on CIFAR10 and CIFAR100. We further provide experiments on ResNet34-TinyImageNet and AlexNet-CIFAR10 in Supplementary G.

Effect of ET Ticket. In Table 1, we report the change in accuracy and search speed gain from applying ET to IMP and EB, on CIFAR10 and CIFAR100 datasets (SVHN and Fashion-MNIST results are provided in Supplementary E). Although IMP achieves highest accuracy across all sparsity levels, they require $26 \sim 73$ hours to search winning tickets with 98.13% sparsity. By applying ET to IMP, the search speed increases up to $\times 1.59$ without a huge accuracy drop (of course, there is an accuracy-computational cost trade-off because the ET winning ticket cannot exactly match with the IMP winning ticket). Compared to IMP, EB ticket provides significantly less search cost for searching winning tickets. Combining ET with EB ticket brings faster search speed, even finding a winning ticket in one hour (on GPU). At high sparsity levels of EB ($p = 98.13\%$), search speed gain from ET is upto $\times 1.16$. Also, applying ET on ResNet19 brings a better speed gain than VGG16 since ResNet19 requires a larger computational graph from multiple timestep operations (details are in Supplementary D). Overall, the results support our hypothesis that important weight connectivity of the SNN can be discovered from shorter timesteps.

Observations from Pruning Techniques. We use *Late Rewinding* [22] (IMP) for obtaining stable performance at high sparsity regime (refer Section 3.1). To analyze the effect of the rewinding epoch, we change the rewinding epoch and report the accuracy on four sparsity levels in Fig. 6(a). We observe that the rewinding epoch does not cause a huge accuracy change with sparsity $\leq 95.69\%$. However, a high sparsity level (98.13%) shows non-trivial accuracy drop $\sim 1.5\%$ at epoch 260, which requires careful rewinding epoch selection. Frankle *et al.* [22] also show that using the same pruning percentage for both shallow and deep layers (*i.e.*, local pruning) degrades the accuracy. Instead of local pruning, they apply different pruning percentages for each layer (*i.e.*, global pruning). We
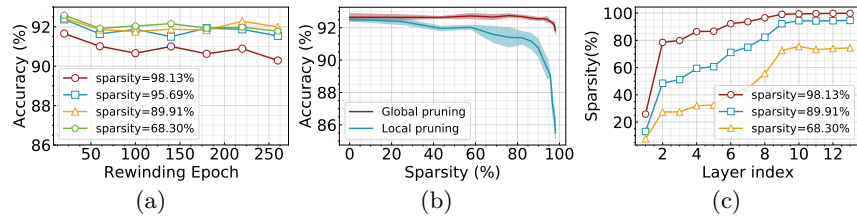
Fig. 6. Observations from Iterative Magnitude Pruning (IMP). (a) The performance change with respect to the rewinding epoch. (b) The performance of global pruning and local pruning. (c) Layer-wise sparsity across different sparsity levels with global pruning. We use VGG16 on the CIFAR10 dataset for experiments.
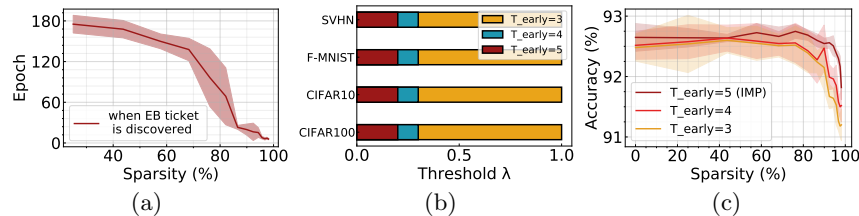

Fig. 7. Observations from Early-Bird (EB) ticket and Early-Time (ET) ticket. (a) Epoch when the EB ticket is discovered. (b) The change of T_{early} with respect to the threshold λ for KL divergence. (c) The performance of winning tickets from different T_{early}. We use VGG16 on the CIFAR10 dataset and show standard deviation from 5 random runs.

compare global pruning and local pruning in Fig. 6(b). In SNN, global pruning achieves better performance than local pruning, especially for high sparsity levels. Fig. 6(c) illustrates layer-wise sparsity obtained from global pruning. The results show that deep layers have higher sparsity compared to shallow layers.

We also visualize the epoch when an EB ticket is discovered with respect to sparsity levels, in Fig. 7(a). The EB ticket obtains a pruning mask based on the mask difference between the current mask and the masks from the previous epochs. Here, we observe that a highly sparse mask is discovered earlier than a lower sparsity mask according to EB ticket's mask detection algorithm (refer Fig. 3 of [80]). Furthermore, we conduct a hyperparameter analysis of the proposed ET ticket. In Fig. 7(b), we show the change of T_{early} with respect to the threshold λ used for selection (Algorithm 1). We search λ with intervals of 0.1 from 0 to 1. Low λ value indicates that we select T_{early} that is similar to the original timestep, and brings less efficiency gain. Interestingly, the trend is similar across different datasets, which indicates our KL divergence works as a consistent metric. Fig. 7(c) shows the accuracy of sparse SNNs from three different T_{early} (Note, $T_{early} = 5$ is the original IMP). The results show that smaller T_{early} also captures important connections in SNNs.
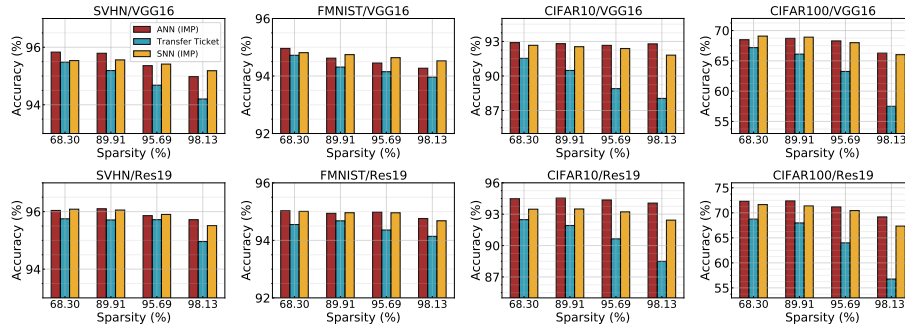
Fig. 8. Transferability study of ANN winning tickets on SNNs.

4.3   Transferred Winning Tickets from ANN

The transferability of winning tickets has been actively explored in order to eliminate search costs. A line of work [50,51,16,7] discover the existence of transferable winning tickets from the source dataset and successfully transfers it to the target dataset. With a different perspective from the prior works which focus on cross-dataset configuration, we discover the transferable winning ticket between ANN and SNN where the activation function is different. In Fig. 8, we illustrate the accuracy of IMP on ANN, IMP on SNN, Transferred Ticket, across four sparsity levels (68.30%, 89.91%, 95.69%, 98.13%). Specifically, Transferred Ticket (*i.e.*, initialized weight parameters and pruning mask) is discovered by IMP on ANN, and trained on SNN framework where we change ReLU neuron to LIF neuron. For relatively simple datasets such as SVHN and F-MNIST, Transferred Ticket shows less than 2% accuracy drop even at 98.13% sparsity. However, for CIFAR10 and CIFAR100, Transferred Ticket fails to detect a winning ticket and shows a huge performance drop. The results show that ANN and SNN share common knowledge, but are not exactly the same, which can be supported by the previous SNN works [62,17,35] where a pretrained ANN provides better initialization for SNN. Although Transferred Ticket shows limited performance than IMP, searching Transferred Ticket from ANN requires ~ 14 hours for 98.13% sparsity, which is $\sim 5\times$ faster than IMP on SNN.

4.4   Finding Winning Tickets from Initialization

A line of work [41,70] effectively reduces search cost for winning tickets by conducting a search process at initialization. This technique should be explored with SNNs where multiple feedforward steps corresponding to multiple timesteps bring expensive search costs. To show this, we conduct experiments on a representative *pruning at initialization* method, SNIP [41]. SNIP computes the importance of each weight connection from the magnitude of backward gradients at initialization. In Fig. 9, we illustrate the accuracy of ANN and SNN with SNIP on VGG16/CIFAR10 configuration.

Table 2. Performance comparison of IMP [21] with the previous works.

| Pruning Method	Architecture	Dataset	Baseline Acc. (%)	ΔAcc (%)	Sparsity (%)
Deng *et al.* [14]	7Conv, 2FC	CIFAR10	89.53	-0.38 -2.16 -3.85	50.00 75.00 90.00
Bellec *et al.* [3]	6Conv, 2FC	CIFAR10	92.84	-1.98 -2.56 -3.53	94.76 98.05 98.96
Chen *et al.* [9]	6Conv, 2FC	CIFAR10	92.84	-0.30 -0.81 -1.47	71.59 94.92 97.65
Chen *et al.* [9]*	ResNet19	CIFAR10	93.22	-0.54 -1.31 -2.10	76.90 94.25 97.56
IMP on SNN (ours)	**ResNet19**	CIFAR10	**93.22**	**+0.28** **+0.24** **-0.04**	**76.20** **94.29** **97.54**
Chen *et al.* [9]*	ResNet19	CIFAR100	71.34	-1.98 -3.87 -4.03	77.03 94.92 97.65
IMP on SNN (ours)	**ResNet19**	CIFAR100	**71.34**	**+0.11** **-0.34** **-2.29**	**76.20** **94.29** **97.54**

* We reimplement ResNet19 experiments.
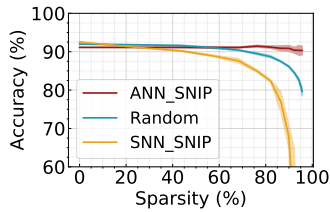
Fig. 9. Performance comparison across pruned ANN and pruned SNN with SNIP.
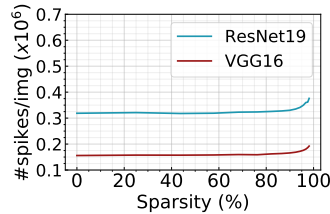

Fig. 10. Number of spikes with respect to sparsity on CIFAR10.

Surprisingly, SNN shows huge performance degradation at high sparsity regime ($> 80\%$), even worse than random pruning. The results imply that the previous *pruning at initialization* based on the backward gradient (tailored for ANNs) is not compatible with SNNs where the backward gradient is approximated because of the non-differentiability of LIF neuron [54,74].

4.5   Performance Comparison with Previous Works

In Table 2, we compare the LTH method, especially IMP, with state-of-the-art SNN pruning works [14,3,9] in terms of accuracy and achieved sparsity. We show the baseline accuracy of the unpruned model and report the accuracy drop at three different sparsity levels. Note that the previous works use a shallow model with $6 \sim 7$ conv and 2 FC layers. The ResNet19 architecture achieves higher baseline accuracy compared to the previous shallow architectures. To compare

the results on a ResNet19 model, we prune ResNet19 with method proposed by Chen *et al.* [9] using the official code[1] provided by authors. We observe that the previous SNN pruning works fail to achieve matching performance at a high sparsity on deeper SNN architectures. On the other hand, IMP shows less performance drop (in some cases even performance improvement) compared to the previous pruning techniques. Especially, at 97.54% sparsity, [9] shows 2.1% accuracy drop whereas IMP degrades the accuracy by 0.04%. Further, we compare the accuracy on the CIFAR100 dataset to explore the effectiveness of the pruning method with respect to complex datasets. Chen *et al.* 's method fails to discover the winning ticket at sparsity 94.29%, and IMP shows better performance across all sparsity levels. The results imply that the LTH-based method might bring a huge advantage as SNNs are scaled up in the future.

4.6   Observation on the Number of Spikes

In general, the energy consumption of SNNs is proportional to the number of spikes [1,13,79] and weight sparsity. In Fig. 10, we measure the number of spikes per image across various sparsity levels of VGG16 and ResNet19 architectures. We use IMP for pruning SNNs. We observe that the sparse SNN maintains a similar number of spikes across all sparsity levels. The results indicate that sparse SNNs obtained with LTH do not bring an additional MAC energy-efficiency gain from spike sparsity. Nonetheless, weight sparsity brings less memory occupation and can alleviate the memory movement overheads [10,58]. As discussed in [6], with over 98% of the sparse weights, SNNs can reduce the memory movement energy by up to 15×.

5   Conclusion

In this work, we aim to explore lottery ticket hypothesis based pruning for implementing sparse deep SNNs at lower search costs. Such an objective is important as SNNs are a promising candidate for deployment on resource-constrained edge devices. To this end, we apply various techniques including Iterative Magnitude Pruning (IMP), Early-Bird ticket (EB), and the proposed Early-Time ticket (ET). Our key observations are summarized as follows: (1) IMP can achieve higher sparsity levels of deep SNN models compared to previous works on SNN pruning. (2) EB ticket reduces search time significantly, but it cannot achieve a winning ticket over 90% sparsity. (3) Adding ET ticket accelerates search speed for both IMP and EB, by up to ×1.66. We can find a winning ticket in one hour with EB+ET, which can enable practical pruning on edge devices.

---

[1] https://github.com/Yanqi-Chen/Gradient-Rewiring

References

1. Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.J., et al.: Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. IEEE transactions on computer-aided design of integrated circuits and systems **34**(10), 1537–1557 (2015)
2. Bai, Y., Wang, H., TAO, Z., Li, K., Fu, Y.: Dual lottery ticket hypothesis. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=fOsN52jn25l
3. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., Maass, W.: Long short-term memory and learning-to-learn in networks of spiking neurons. Advances in neural information processing systems **31** (2018)
4. Brix, C., Bahar, P., Ney, H.: Successfully applying the stabilized lottery ticket hypothesis to the transformer architecture. arXiv preprint arXiv:2005.03454 (2020)
5. Burkholz, R., Laha, N., Mukherjee, R., Gotovos, A.: On the existence of universal lottery tickets. arXiv preprint arXiv:2111.11146 (2021)
6. Chen, G.K., Kumar, R., Sumbul, H.E., Knag, P.C., Krishnamurthy, R.K.: A 4096-neuron 1m-synapse 3.8-pj/sop spiking neural network with on-chip stdp learning and sparse weights in 10-nm finfet cmos. IEEE Journal of Solid-State Circuits **54**(4), 992–1002 (2018)
7. Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., Carbin, M.: The lottery ticket hypothesis for pre-trained bert networks. Advances in neural information processing systems **33**, 15834–15846 (2020)
8. Chen, T., Zhang, Z., pengjun wang, Balachandra, S., Ma, H., Wang, Z., Wang, Z.: Sparsity winning twice: Better robust generalization from more efficient training. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=SYuJXrXq8tw
9. Chen, Y., Yu, Z., Fang, W., Huang, T., Tian, Y.: Pruning of deep spiking neural networks through gradient rewiring. arXiv preprint arXiv:2105.04916 (2021)
10. Chen, Y.H., Emer, J., Sze, V.: Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. ACM SIGARCH Computer Architecture News **44**(3), 367–379 (2016)
11. Christensen, D.V., Dittmann, R., Linares-Barranco, B., Sebastian, A., Le Gallo, M., Redaelli, A., Slesazeck, S., Mikolajick, T., Spiga, S., Menzel, S., et al.: 2022 roadmap on neuromorphic computing and engineering. Neuromorphic Computing and Engineering (2022)
12. Comsa, I.M., Fischbacher, T., Potempa, K., Gesmundo, A., Versari, L., Alakuijala, J.: Temporal coding in spiking neural networks with alpha synaptic function. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 8529–8533. IEEE (2020)
13. Davies, M., Srinivasa, N., Lin, T.H., Chinya, G., Cao, Y., Choday, S.H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al.: Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro **38**(1), 82–99 (2018)
14. Deng, L., Wu, Y., Hu, Y., Liang, L., Li, G., Hu, X., Ding, Y., Li, P., Xie, Y.: Comprehensive snn compression using admm optimization and activity regularization. IEEE transactions on neural networks and learning systems (2021)
15. Deng, S., Li, Y., Zhang, S., Gu, S.: Temporal efficient training of spiking neural network via gradient re-weighting. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=_XNtisL32jv

16. Desai, S., Zhan, H., Aly, A.: Evaluating lottery tickets under distributional shifts. arXiv preprint arXiv:1910.12708 (2019)
17. Ding, J., Yu, Z., Tian, Y., Huang, T.: Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. arXiv preprint arXiv:2105.11654 (2021)
18. Ding, S., Chen, T., Wang, Z.: Audio lottery: Speech recognition made ultra-lightweight, noise-robust, and transferable. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=9Nk6AJkVYB
19. Fang, W., Chen, Y., Ding, J., Chen, D., Yu, Z., Zhou, H., Tian, Y., other contributors: Spikingjelly. https://github.com/fangwei123456/spikingjelly (2020)
20. Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., Tian, Y.: Deep residual learning in spiking neural networks. Advances in Neural Information Processing Systems **34** (2021)
21. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635 (2018)
22. Frankle, J., Dziugaite, G.K., Roy, D.M., Carbin, M.: Stabilizing the lottery ticket hypothesis. arXiv preprint arXiv:1903.01611 (2019)
23. Furber, S.B., Galluppi, F., Temple, S., Plana, L.A.: The spinnaker project. Proceedings of the IEEE **102**(5), 652–665 (2014)
24. Girish, S., Maiya, S.R., Gupta, K., Chen, H., Davis, L.S., Shrivastava, A.: The lottery ticket hypothesis for object recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 762–771 (2021)
25. Guo, W., Fouda, M.E., Yantir, H.E., Eltawil, A.M., Salama, K.N.: Unsupervised adaptive weight pruning for energy-efficient neuromorphic systems. Frontiers in Neuroscience p. 1189 (2020)
26. Han, S., Pool, J., Narang, S., Mao, H., Gong, E., Tang, S., Elsen, E., Vajda, P., Paluri, M., Tran, J., et al.: Dsd: Dense-sparse-dense training for deep neural networks. arXiv preprint arXiv:1607.04381 (2016)
27. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. Advances in neural information processing systems **28** (2015)
28. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
29. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
30. Izhikevich, E.M.: Simple model of spiking neurons. IEEE Transactions on neural networks **14**(6), 1569–1572 (2003)
31. Kalibhat, N.M., Balaji, Y., Feizi, S.: Winning lottery tickets in deep generative models. arXiv preprint arXiv:2010.02350 (2020)
32. Kim, Y., Li, Y., Park, H., Venkatesha, Y., Panda, P.: Neural architecture search for spiking neural networks. arXiv preprint arXiv:2201.10355 (2022)
33. Kim, Y., Panda, P.: Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. Frontiers in neuroscience p. 1638 (2020)
34. Kim, Y., Panda, P.: Visual explanations from spiking neural networks using interspike intervals. Sci Rep 11, 19037 (2021). https://doi.org/10.1038/s41598-021-98448-0 (2021)
35. Kim, Y., Venkatesha, Y., Panda, P.: Privatesnn: Privacy-preserving spiking neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 1192–1200 (2022)
36. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

37. Kundu, S., Pedram, M., Beerel, P.A.: 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. pp. 5209–5218 (2021)

38. Ledinauskas, E., Ruseckas, J., Juršėnas, A., Buračas, G.: Training deep spiking neural networks. arXiv preprint arXiv:2006.04436 (2020)

39. Lee, C., Sarwar, S.S., Panda, P., Srinivasan, G., Roy, K.: Enabling spike-based backpropagation for training deep neural network architectures. Frontiers in Neuroscience **14** (2020)

40. Lee, J.H., Delbruck, T., Pfeiffer, M.: Training deep spiking neural networks using backpropagation. Frontiers in neuroscience **10**, 508 (2016)

41. Lee, N., Ajanthan, T., Torr, P.H.: Snip: Single-shot network pruning based on connection sensitivity. arXiv preprint arXiv:1810.02340 (2018)

42. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)

43. Li, Y., Deng, S., Dong, X., Gong, R., Gu, S.: A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. arXiv preprint arXiv:2106.06984 (2021)

44. Li, Y., Deng, S., Dong, X., Gu, S.: Converting artificial neural networks to spiking neural networks via parameter calibration. arXiv preprint arXiv:2205.10121 (2022)

45. Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., Gu, S.: Differentiable spike: Rethinking gradient-descent for training spiking neural networks. Advances in Neural Information Processing Systems **34** (2021)

46. Liu, S., Chen, T., Atashgahi, Z., Chen, X., Sokar, G., Mocanu, E., Pechenizkiy, M., Wang, Z., Mocanu, D.C.: Deep ensembling with no overhead for either training or testing: The all-round blessings of dynamic sparsity. arXiv preprint arXiv:2106.14568 (2021)

47. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning. arXiv preprint arXiv:1810.05270 (2018)

48. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)

49. Martinelli, F., Dellaferrera, G., Mainar, P., Cernak, M.: Spiking neural networks trained with backpropagation for low power neuromorphic implementation of voice activity detection. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 8544–8548. IEEE (2020)

50. Mehta, R.: Sparse transfer learning via winning lottery tickets. arXiv preprint arXiv:1905.07785 (2019)

51. Morcos, A., Yu, H., Paganini, M., Tian, Y.: One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. Advances in neural information processing systems **32** (2019)

52. Mostafa, H.: Supervised learning based on temporal coding in spiking neural networks. IEEE transactions on neural networks and learning systems **29**(7), 3227–3235 (2017)

53. Movva, R., Zhao, J.Y.: Dissecting lottery ticket transformers: Structural and behavioral study of sparse neural machine translation. arXiv preprint arXiv:2009.13270 (2020)

54. Neftci, E.O., Mostafa, H., Zenke, F.: Surrogate gradient learning in spiking neural networks. IEEE Signal Processing Magazine **36**, 61–63 (2019)

55. Neftci, E.O., Pedroni, B.U., Joshi, S., Al-Shedivat, M., Cauwenberghs, G.: Stochastic synapses enable efficient brain-inspired learning machines. Frontiers in neuroscience **10**, 241 (2016)

56. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
57. Orchard, G., Frady, E.P., Rubin, D.B.D., Sanborn, S., Shrestha, S.B., Sommer, F.T., Davies, M.: Efficient neuromorphic signal processing with loihi 2. In: 2021 IEEE Workshop on Signal Processing Systems (SiPS). pp. 254–259. IEEE (2021)
58. Parashar, A., Rhu, M., Mukkara, A., Puglielli, A., Venkatesan, R., Khailany, B., Emer, J., Keckler, S.W., Dally, W.J.: Scnn: An accelerator for compressed-sparse convolutional neural networks. ACM SIGARCH computer architecture news **45**(2), 27–40 (2017)
59. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: NIPS-W (2017)
60. Rathi, N., Panda, P., Roy, K.: Stdp-based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **38**(4), 668–677 (2018)
61. Rathi, N., Roy, K.: 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)
62. Rathi, N., Srinivasan, G., Panda, P., Roy, K.: Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. arXiv preprint arXiv:2005.01807 (2020)
63. Roy, K., Jaiswal, A., Panda, P.: Towards spike-based machine intelligence with neuromorphic computing. Nature **575**(7784), 607–617 (2019)
64. Schuman, C.D., Kulkarni, S.R., Parsa, M., Mitchell, J.P., Kay, B., et al.: Opportunities for neuromorphic computing algorithms and applications. Nature Computational Science **2**(1), 10–19 (2022)
65. Shi, Y., Nguyen, L., Oh, S., Liu, X., Kuzum, D.: A soft-pruning method applied during training of spiking neural networks for in-memory computing applications. Frontiers in neuroscience **13**, 405 (2019)
66. Shrestha, S.B., Orchard, G.: Slayer: Spike layer error reassignment in time. arXiv preprint arXiv:1810.08646 (2018)
67. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ICLR (2015)
68. Venkatesha, Y., Kim, Y., Tassiulas, L., Panda, P.: Federated learning with spiking neural networks. arXiv preprint arXiv:2106.06579 (2021)
69. Vischer, M.A., Lange, R.T., Sprekeler, H.: On lottery tickets and minimal task representations in deep reinforcement learning. arXiv preprint arXiv:2105.01648 (2021)
70. Wang, C., Zhang, G., Grosse, R.: Picking winning tickets before training by preserving gradient flow. arXiv preprint arXiv:2002.07376 (2020)
71. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. Advances in neural information processing systems **29** (2016)
72. Wu, H., Zhang, Y., Weng, W., Zhang, Y., Xiong, Z., Zha, Z.J., Sun, X., Wu, F.: Training spiking neural networks with accumulated spiking flow. ijo **1**(1) (2021)
73. Wu, J., Xu, C., Zhou, D., Li, H., Tan, K.C.: Progressive tandem learning for pattern recognition with deep spiking neural networks. arXiv preprint arXiv:2007.01204 (2020)
74. Wu, Y., Deng, L., Li, G., Zhu, J., Shi, L.: Spatio-temporal backpropagation for training high-performance spiking neural networks. Frontiers in neuroscience **12**, 331 (2018)

75. Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., Shi, L.: Direct training for spiking neural networks: Faster, larger, better. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 1311–1318 (2019)
76. Wu, Y., Zhao, R., Zhu, J., Chen, F., Xu, M., Li, G., Song, S., Deng, L., Wang, G., Zheng, H., et al.: Brain-inspired global-local learning incorporated with neuromorphic computing. Nature Communications **13**(1), 1–14 (2022)
77. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
78. Yao, M., Gao, H., Zhao, G., Wang, D., Lin, Y., Yang, Z., Li, G.: Temporal-wise attention spiking neural networks for event streams classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10221–10230 (2021)
79. Yin, R., Moitra, A., Bhattacharjee, A., Kim, Y., Panda, P.: Sata: Sparsity-aware training accelerator for spiking neural networks. arXiv preprint arXiv:2204.05422 (2022)
80. You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R.G., Wang, Z., Lin, Y.: Drawing early-bird tickets: Towards more efficient training of deep networks. arXiv preprint arXiv:1909.11957 (2019)
81. Yu, H., Edunov, S., Tian, Y., Morcos, A.S.: Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. arXiv preprint arXiv:1906.02768 (2019)
82. Zhang, Z., Chen, X., Chen, T., Wang, Z.: Efficient lottery ticket finding: Less data is more. In: International Conference on Machine Learning. pp. 12380–12390. PMLR (2021)
83. Zheng, H., Wu, Y., Deng, L., Hu, Y., Li, G.: Going deeper with directly-trained larger spiking neural networks. arXiv preprint arXiv:2011.05280 (2020)
84. Zhou, H., Lan, J., Liu, R., Yosinski, J.: Deconstructing lottery tickets: Zeros, signs, and the supermask. Advances in neural information processing systems **32** (2019)