State Transition of Dendritic Spines Improves Learning of Sparse Spiking Neural Networks

Yanqi Chen [1 2] **Zhaofei Yu** [† 1 2 3] **Wei Fang** [1 2] **Zhengyu Ma** [† 2] **Tiejun Huang** [1 2 3] **Yonghong Tian** [† 1 2]

Abstract

Spiking Neural Networks (SNNs) are considered a promising alternative to Artificial Neural Networks (ANNs) for their event-driven computing paradigm when deployed on energy-efficient neuromorphic hardware. Recently, deep SNNs have shown breathtaking performance improvement through cutting-edge training strategy and flexible structure, which also scales up the number of parameters and computational burdens in a single network. Inspired by the state transition of dendritic spines in the filopodial model of spinogenesis, we model different states of SNN weights, facilitating weight optimization for pruning. Furthermore, the pruning speed can be regulated by using different functions describing the growing threshold of state transition. We organize these techniques as a dynamic pruning algorithm based on nonlinear reparameterization mapping from spine size to SNN weights. Our approach yields sparse deep networks on the large-scale dataset (SEW ResNet18 on ImageNet) while maintaining state-of-the-art low performance loss ($\sim$3% at 88.8% sparsity) compared to existing pruning methods on directly trained SNNs. Moreover, we find out pruning speed regulation while learning is crucial to avoiding disastrous performance degradation at the final stages of training, which may shed light on future work on SNN pruning.

1. Introduction

Spiking neural networks (SNNs), acclaimed as the third generation of neural network models (Maass, 1997), have drawn more and more attention from researchers in the past few years. Compared to artificial neural networks (ANNs), SNNs go a step further in mimicking the computing pattern of human brains. Neurons in SNNs release discrete events, namely "spikes", to communicate with each other and thereby called spiking neurons. As human brains consume only around 20W (Mink et al., 1981) while processing various complicated tasks, SNNs are also energy efficient when deployed on neuromorphic hardwares (Merolla et al., 2014; Furber et al., 2014; Ma et al., 2017; Davies et al., 2018; Pei et al., 2019). Furthermore, the spiking neuron models follow their biological counterparts and inherit complex temporal dynamics from them, endowing SNNs with a strong ability to extract spatio-temporal features on multiple tasks such as recognition (Payeur et al., 2021; Stöckl & Maass, 2021; Wu et al., 2021; Fang et al., 2021a;b), detection (Kim et al., 2020), segmentation (Kirkland et al., 2020), and tracking (Luo et al., 2019; 2021).

Larger and deeper SNNs are demanded to handle tasks with growing scales of datasets. In the field of ANNs, large-scale datasets like ImageNet require networks with an enormous number of parameters, such as GoogLeNet (Szegedy et al., 2015) or ResNet (He et al., 2016). The situation for SNNs is similar, thereby large SNNs with the same structures like Spiking ResNet (Zheng et al., 2021; Hu et al., 2021) and SpikingGoogleNet (Zhou et al., 2021) are built. However, the number of parameters in these architectures is far beyond what neuromorphic chips can now support, especially for those with one hundred or more layers. For example, a 48-node SpiNNaker board is able to simulate networks with up to 250,000 neurons and 80 million synapses (Furber et al., 2014), which is not enough for any single network mentioned above. Therefore, pruning redundant weights in SNNs is a promising approach to deploy large networks to event-based hardware.

The sparsification of network weights, or pruning of networks has been a flourishing area (Han et al., 2015) since deep ANNs show their power on real-life tasks. Compared with ANNs, SNNs make better use of unstructured sparsity for their event-driven nature. This is because the computation on event-driven hardware is triggered only when both incoming spikes and network weights are nonzero (Merolla

[1]National Engineering Research Center of Visual Technology, School of Computer Science, Peking University [2]Peng Cheng Laboratory [3]Institute for Artificial Intelligence, Peking University. Correspondence to: Zhaofei Yu <yuzf12@pku.edu.cn>, Zhengyu Ma <mazhy@pcl.ac.cn>, Yonghong Tian <yhtian@pku.edu.cn>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

et al., 2014), which does not require coarse-grained sparsity to reduce running costs as in ANNs. Based on the above reason, we need a feasible unstructured pruning algorithm. In previous studies of SNN pruning, researchers show a preference for existing methods from ANN pruning. These methods work well on smaller datasets like MNIST or CIFAR-10, whereas they have not been evaluated on large-scale networks or complex datasets like ImageNet. Some studies turn to biological mechanisms and propose heuristic methods. However, these techniques are more or less confined to biological facts and hence lack rationality from the perspective of sparse optimization.

In this paper, we develop a reparameterization framework from the size of the spine to the real connection weights based on the state transition of the dendritic spine in the neural system, enabling us to prune connections of SNNs. Our modeling facilitates the sparsification of deep SNNs while maintaining low performance loss, reaching the state-of-the-art balance between accuracy and sparsity on large-scale datasets. Furthermore, we show the importance of scheduling threshold between the state "mature spine" and the state "filopodium" by extensive experiments on the performance of sparse SNNs.

The main contributions of this paper can be summarized as follows:

- Inspired by the filopodial model of spinogenesis, we manage to model two kinds of state transitions of dendritic spines: 1) Mature dendritic spines and filopodia 2) Excitatory and inhibitory synapses; by using a non-linear reparameterization function. It gets rid of Dale's law and keeps exchanging the signs of weights for efficiency optimization, while still maintaining the ability to sparsify weights of SNNs.

- We provide proof of convergence and theoretical analysis of the sparsity springhead in our algorithm from the perspective of sparse optimization.

- We conduct extensive experiments on pruning of SEW ResNet18 on ImageNet classification tasks and it achieves state-of-the-art low performance loss. To the best of our knowledge, it is the first time highly sparse SNNs (88.8% sparsity) achieve so little performance loss ($\sim 3\%$) on ImageNet dataset.

- We make an in-depth discussion of excellence and disadvantages between different settings (threshold schedulers) of our approach by tracking multiple properties during learning. We also show a preliminary explanation of the performance variation.

2. Related Work

Previous researches on lightweight SNNs are principally conducted from five aspects: 1) Pruning 2) Quantization 3) Reduction of firing rates 4) Sparse training 5) Knowledge distillation.

Quantization methods for SNNs have been developed for a long time since the same techniques were applied to ANNs (Stromatias et al., 2015; Neftci et al., 2016). Recent advances in quantization for SNNs (Qiao et al., 2021; Wang et al., 2021; Kheradpisheh et al., 2021) are mainly focused on binary weights and different tasks. Introducing firing rate as an explicit regularization term to the loss is the most general way to suppress firing rates (Neil et al., 2016; Zenke & Vogels, 2021; Deng et al., 2021). Sparse approximation to gradients reduces training costs of SNNs (Zenke & Neftci, 2021; Perez-Nieves & Goodman, 2021). Distilling knowledge from large ANN (Takuya et al., 2021) or SNN (Kushawaha et al., 2021) to smaller SNN is also a substitute for direct compression.

By comparison, the pruning for SNNs draws more attention since it has corresponding physiological processes, which leads a way to efficient SNN structure. To the best of our knowledge, the earliest series of researches concerning pruning of networks with firing neurons (Chechik et al., 1998; 1999) borrows biological mechanisms like synaptic turnover.

Recent researches on SNN pruning are primarily from two starting points: 1) Learning from successful experience of ANN pruning 2) Exploring biological counterparts. The former technical route is much more popular, which initially enables some tentative work inserting magnitude-based pruning to different training methods of SNNs, e.g., *event-driven contrastive divergence* (eCD) (Neftci et al., 2016), STDP (Rathi et al., 2019; Nguyen et al., 2021), ANN2SNN (Mern et al., 2017; Liu et al., 2020). The success of training deep SNNs using the surrogate gradient method (Neftci et al., 2019), or STBP (Wu et al., 2018), arouses a swarm of pruning strategies (Martinelli et al., 2020; Deng et al., 2021; Yin et al., 2021) via integrating newly developed ANN pruning algorithms. Meanwhile, in view of the similarity between SNNs and neural systems, there are some attempts to bio-inspired pruning algorithms. Modeling the regrowth process (Kundu et al., 2021) is proved to be a competent strategy. This idea can be further combined with the modeling of dendritic spines in SNNs to enhance pruning, which introduces spine motility (Kappel et al., 2015; Bellec et al., 2018b) or gradient (Chen et al., 2021) as a criterion of regrowth. Both criteria achieve a reasonable trade-off between sparsity and accuracy on deep SNNs and also provide solid theoretical proof for their algorithms.

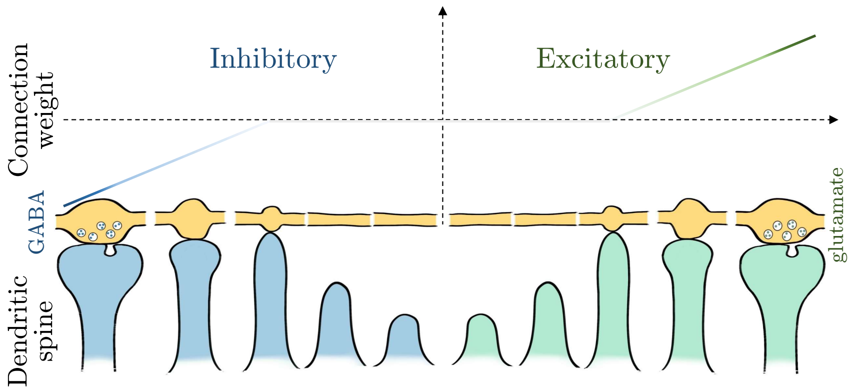

Figure 1. Scheme for weight reparameterization and transitions between filopodia and mature dendritic spines, and transitions between different types of synapses. Yellow areas represent the axon terminal. Blue and green areas distinguish two types of dendritic spines, i.e. inhibitory and excitatory. GABA and glutamate exemplify the main kinds of inhibitory and excitatory neurotransmitters in the mammalian cortex respectively.

3. Preliminaries

3.1. Spiking Neuron Model

The *Leaky Integrate-and-Fire (LIF)* model (Gerstner & Kistler, 2002) is among the most popular neuron models for its low computing costs. For a layer of spiking neurons, we denote $I_i(t)$ as the input from the i-th presynaptic neuron and w_i as the corresponding synaptic weights. The subthreshold membrane potential $u(t)$ of the postsynaptic neuron can be expressed in the form of an ordinary differential equation (ODE).

$$\tau_m \frac{\mathrm{d}u(t)}{\mathrm{d}t} = -(u(t) - u_{\text{rest}}) + \sum\nolimits_i w_i I_i(t), \tag{1}$$

where τ_m is a positive membrane time constant, and u_{rest} is the resting potential. When $u(t)$ reaches the firing threshold u_{th} at time t^f, a spike is generated and at the same time, the membrane potential $u(t)$ is reset to the resting potential u_{rest}.

$$\lim_{\Delta t \to 0^+} u(t^f + \Delta t) = u_{\text{rest}}, \text{ if } u(t^f) \geq u_{\text{th}}. \tag{2}$$

In computer simulation, we generally need to consider a discrete version of Eq. (1) and Eq. (2). By Euler method, Eq. (1) and Eq. (2) can be discretized as follows:

$$\begin{aligned} u[t^-] &= u[t-1] \\ &\quad + \frac{1}{\tau_m}\left(-(u[t-1] - u_{\text{rest}}) + \sum\nolimits_i w_i I_i[t]\right), \\ s[t] &= H(u[t^-] - u_{\text{th}}), \\ u[t] &= s[t]u_{\text{rest}} + (1 - s[t])u[t^-], \end{aligned} \tag{3}$$

where $u[t^-], u[t]$ are the membrane potential before and after firing at timestep t respectively, and $H(\cdot)$ is the Heaviside step function. If $u[t^-] \geq u_{\text{th}}$, it will produce a binary spike $s[t]$ and the $u[t]$ will reset to u_{rest} or otherwise left unchanged.

3.2. Dendritic Spine

Dendritic spines (or simply *spines*) are small membranous protrusions that arise from dendrites, which connect dendrites to axon terminals (Yuste, 2010), allowing presynaptic spikes to be delivered to postsynaptic neurons. Fig. 1 illustrates the status of contact between spines, which has two types (*excitatory* and *inhibitory*) and various sizes, and axons. It is widely recognized that synaptic weights in the mammalian brain are significantly correlated to the size factor, e.g., head size (Matsuzaki et al., 2001; 2004; Béïque et al., 2006), neck length (Noguchi et al., 2005) and neck width (Tønnesen et al., 2014) of spines, which affects the long-term dynamics like long-term potentiation (LTP) and long-term depression (LTD) (Engert & Bonhoeffer, 1999). Larger spines have more receptors and thus inject more current into the dendrite, arousing stronger inhibitory postsynaptic potential (IPSP) or excitatory postsynaptic potential (EPSP). The type of synapse is determined by neurotransmitters released from axon terminals of presynaptic neurons and receptors on spines. As shown in Fig. 1, glutamate and gamma-aminobutyric acid (GABA) are, respectively, the major excitatory and the major inhibitory neurotransmitters in the mature mammalian brain (Fonnum, 1984; Bhat et al., 2010). Spikes from excitatory synapses increase the firing rates of neurons while spikes from inhibitory synapses do the opposite, which corresponds to the positive and negative weights in SNNs, respectively. Weight changes in SNNs are similar and usually considered analogical to enlargement or shrinkage of spines (Kappel et al., 2015).

4. Methodology

4.1. Reparameterization of Weights

The synaptic weights in SNNs are generally treated as their counterparts in ANNs. Despite the previous success in deep learning of SNNs, there are few attempts to model pruning in SNNs as a gradient-based optimization problem since it is hard to restrict the synaptic weights exactly to zero, where it generally has a nonzero gradient. We point out that the status "pruned" should not be modeled as a single point but an interval on the real axis. Furthermore, it is wise to allow the exchange of positive (excitatory) and negative (inhibitory) weights, since they may be initialized improperly if the types are fixed.

In fact, researches in neuroscience have revealed that there exists an intermediate state, namely "dendritic filopodia", for renascent spines in hippocampal neurons (Fiala et al., 1998). Filopodia are deemed the structural precursors of dendritic spines (Zuo et al., 2005), and they rarely form connections to presynaptic axons. We model such behavior as reparameterization mapping from the size of the spine/filopodium $|\theta|$ to the real connection weights w, the sign of weights represents the type of synapses (excitatory/inhibitory), which is formulated as

$$w = \operatorname{sign}(\theta) \cdot (|\theta| - d)_+, d \geq 0, \tag{4}$$

where $(x)_+ := \max(x, 0)$. When the size of the connection is below threshold d, it is considered a filopodium, and the equivalent weight is zero. As shown in Fig. 1, filopodia lack synaptic contact to axons, since they are not large enough to form links. It becomes an existent connection when the size increases to above d. During each update of the weights, the threshold d will increase gradually and turn mature spines to filopodia progressively. Specifically, there are two kinds of state transitions: (1) Transition between filopodia and mature spines (pruning and regrowth), (2) Transition between excitatory and inhibitory synapses (flip sign). A scheme for the weight reparameterization and corresponding state of the spine is shown in Fig. 1. It should be noted that our approach will not change the default initialization of weights. In practice, the weight $\boldsymbol{w}$ is first initialized and then we calculate corresponding spine size $\boldsymbol{\theta}$.

Notice that similar modeling has also been proposed in a series of works of rewiring in ANNs/SNNs. The existing work adheres to Dale's law that the sign of weight (type of neurotransmitter) should not change during learning, i.e., $d \to +\infty$ and fixed (Bellec et al., 2018a; Chen et al., 2021) or using exponential reparameterization (Kappel et al., 2015), which is true for biological synapses. However, these settings are inflexible and introduce unnecessary restrictions to the representation capacity of SNNs. The form of reparameterization function in Eq. (4) is consistent with the model in a previous work (Kusupati et al., 2020) with trainable d concerning ANN pruning. Instead, we focus on finding the best evolution strategy of d, rather than giving an optimal choice of fixed d. Besides, our algorithm is biology-inspired from a completely different perspective.

4.2. Direct Training using Surrogate Gradient

With the proposed reparameterization framework (Eq. (4)), the gradient descent algorithm can be directly applied to optimize the spine size $\boldsymbol{\theta}$ and synaptic weights $\boldsymbol{w}$ of the SNN. Specifically, we manage to derive the gradients of loss function $\mathcal{L}$ with respect to SNN weights $\nabla_{\boldsymbol{w}}\mathcal{L}$ by applying backpropagation through time (BPTT) along the path in the computational graph defined by iterative computing in Eq. (3). Additionally, the gradient with respect to spine size $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ is given through reparameterization mapping by

$$\nabla_{\boldsymbol{\theta}}\mathcal{L} = \nabla_{\boldsymbol{w}}\mathcal{L} \cdot \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{\theta}}. \tag{5}$$

Note that unlike in ANNs, gradient based training methods in SNNs must deal with nondifferentiable Heaviside function $H(\cdot)$ in Eq. (3). The actual derivative of Heaviside function is undefined at zero and 0 elsewhere. The pioneers of SNN learning (Zenke & Ganguli, 2018; Wu et al., 2018; Neftci et al., 2019) introduce some differentiable functions as alternatives of $H(\cdot)$, namely surrogate function, making calculation of gradients tractable. We choose a recently proposed function (Fang et al., 2021a)

$$H^*(x) = \frac{1}{\pi}\arctan(\frac{\pi}{2}\beta x) + \frac{1}{2} \tag{6}$$

as surrogate function during training, where β is the slope parameter.

4.3. Case Study: How sparsity is induced?

To clarify how this algorithm enforces sparsity in SNN learning, we discuss three different cases of update rules. If not otherwise specified, the superscript t here, e.g. d^t, denotes the corresponding symbol after the t-th training step. For convenience, we define the notation of element-wise soft threshold operator as

$$\mathcal{S}_\lambda(x) = \operatorname{sign}(x)\cdot(|x|-\lambda)_+ = \begin{cases} x - \operatorname{sign}(x)\cdot\lambda, & |x| > \lambda \\ 0, & |x| \leq \lambda \end{cases} \tag{7}$$

To begin with, we derive the actual gradient with Eq. (4) and Eq. (5).

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{t-1}) = \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}^{t-1}) \cdot \operatorname{diag}(\mathbf{1}_{|\theta^{t-1}|>d}), \tag{8}$$

where $\mathbf{1}_{|\theta^{t-1}|>d}$ is an indicator vector, of which the corresponding element is 0 when $\theta_i^{t-1} \in [-d, d]$ and 1 otherwise. The general update rule for spine sizes in SNN is given as

$$\begin{aligned} \boldsymbol{\theta}^t &= \boldsymbol{\theta}^{t-1} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{t-1}) \\ &= \boldsymbol{\theta}^{t-1} - \eta\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}^{t-1}) \cdot \operatorname{diag}(\mathbf{1}_{|\theta^{t-1}|>d}), \end{aligned} \tag{9}$$

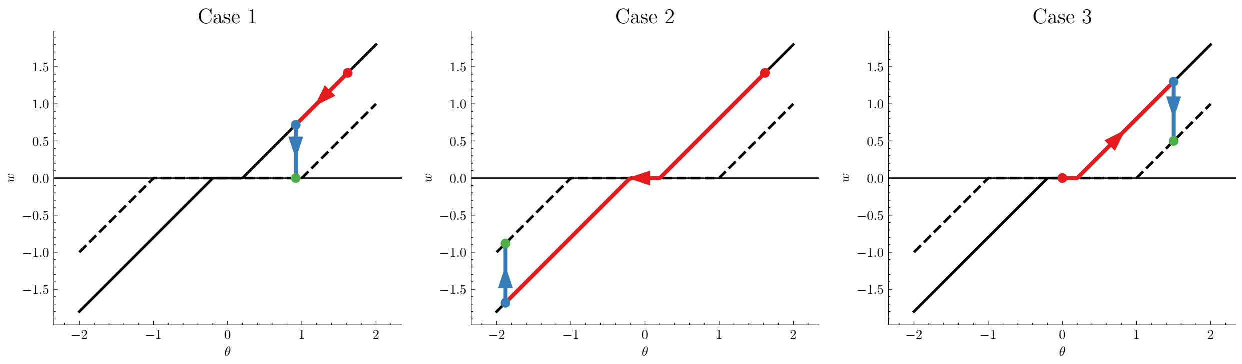

Figure 2. Illustration of different cases discussed in Section 4.3. The red points represents the starting points (θ^{t-1}, w^{t-1}). They move to blue points (θ^t, w^{t^-}) according to gradient update. The increment from d^{t-1} to d^t shrinks the weights and moves the blue points to green points (θ^t, w^t). The solid and dashed black lines represent mapping $\mathcal{S}_{d^{t-1}}(\cdot)$ and $\mathcal{S}_{d^t}(\cdot)$ respectively.

where η is the learning rate. The corresponding gradient update rule for weights is

$$\boldsymbol{w}^{t^-} = \mathcal{S}_{d^{t-1}}(\boldsymbol{\theta}^t), \tag{10}$$

where $\boldsymbol{w}^{t^-}$ is the intermediate weights before shrinkage (update of the threshold d). The above update is followed by the increment of d, which is

$$\boldsymbol{w}^t = \mathcal{S}_{d^t}(\boldsymbol{\theta}^t) = \mathcal{S}_{\Delta d^t}(\boldsymbol{w}^{t^-}) = \mathcal{S}_{\Delta d^t}(\mathcal{S}_{d^{t-1}}(\boldsymbol{\theta}^t)), \tag{11}$$

where $\Delta d^t := d^t - d^{t-1} > 0$.

We classify different cases according to whether an element w^{t-1} of $\boldsymbol{w}^{t-1}$ changes its sign, i.e. $\mathrm{sign}(w^{t^-} w^{t-1}) = 1$ or -1 after gradient update described in Eq. (10), and whether the weight w^{t-1} is zero or not. For a nonzero weight w, the corresponding spine size θ satisfy $|\theta| > d$. Hence, Eq. (10) can be rewritten to:

$$w^{t^-} = \mathcal{S}_{d^{t-1}}(\theta^{t-1} - \eta \nabla_w \mathcal{L}(w^{t-1})). \tag{12}$$

Notice that a trivial case is $w^{t^-} = 0$. In this case, the increment of d has no influence on w. Other cases of nonzero weight are shown in the first two cases below.

Case 1: Same sign, nonzero weight. When the sign of nonzero weight w does not change after applying gradient update, that is, $\mathrm{sign}(\theta^t) = \mathrm{sign}(\theta^{t-1})$ and $|\theta^{t-1}|, |\theta^t| > d$. As shown in Fig. 2, this usually happens when the step size of gradient update is relatively small. By definition in Eq. (7) and Eq. (12), we have

$$\begin{aligned} w^{t^-} &= \theta^{t-1} - \eta \nabla_w \mathcal{L}(w^{t-1}) - \mathrm{sign}(\theta^t) d^{t-1} \\ &= (\theta^{t-1} - \mathrm{sign}(\theta^{t-1}) d^{t-1}) - \eta \nabla_w \mathcal{L}(w^{t-1}) \\ &= w^{t-1} - \eta \nabla_w \mathcal{L}(w^{t-1}). \end{aligned} \tag{13}$$

The last equality holds the nonzero wight assumption and Eq. (4). By combining Eq. (11) and Eq. (13), the overall update rule of weight has the form

$$w^t = \mathcal{S}_{\Delta d^t}\left(w^{t-1} - \eta \nabla_w \mathcal{L}(w^{t-1})\right). \tag{14}$$

This update rule is equivalent to the *Iterative Shrinkage and Thresholding Algorithm* (ISTA) (Daubechies et al., 2004), which iteratively acquires the closed-form solution of the problem

$$\begin{aligned} \boldsymbol{w}^t = &\operatorname*{argmin}_{\boldsymbol{w}} \\ &\left\{ \frac{1}{2\eta} \|\boldsymbol{w} - (\boldsymbol{w}^{t-1} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}^{t-1}))\|_2^2 + \Delta d^t \|\boldsymbol{w}\|_1 \right\}. \end{aligned} \tag{15}$$

It evaluates the proximal gradient under ℓ_1-based regularization $\Delta d^t \|\boldsymbol{w}\|_1$ and induces sparsity. A well-known conclusion (Nesterov, 2007) is that assuming loss function $\mathcal{L}$ is L-smooth and convex, the ISTA has sublinear convergence rate for constant step size $\eta \in (0, 1/L]$.

Further, if d increases linearly, Δd^t is a constant and thereby the regularization term is invariant during learning.

Case 2: Altered sign, nonzero weight. If the step is sufficiently large to change the sign of a weight after gradient update, i.e., $\mathrm{sign}(\theta^t) = -\mathrm{sign}(\theta^{t-1})$ and $|\theta^{t-1}|, |\theta^t| > d$, which implies a alteration of synapse type shown in Fig. 2. By definition in Eq. (7) and Eq. (12), we have

$$\begin{aligned} w^{t^-} &= \theta^{t-1} - \eta \nabla_w \mathcal{L}(w^{t-1}) - \mathrm{sign}(\theta^t) d^{t-1} \\ &= \theta^{t-1} - \eta \nabla_w \mathcal{L}(w^{t-1}) - (\mathrm{sign}(\theta^{t-1}) \\ &\quad + 2\mathrm{sign}(\theta^t)) d^{t-1} \\ &= (\theta^{t-1} - \mathrm{sign}(\theta^{t-1}) d^{t-1}) - \eta \nabla_w \mathcal{L}(w^{t-1}) \\ &\quad - 2\mathrm{sign}(\theta^t) d^{t-1} \\ &= \mathcal{S}_{2d^{t-1}}(w^{t-1} - \eta \nabla_w \mathcal{L}(w^{t-1})). \end{aligned} \tag{16}$$

By Eq. (11), we have

$$w^t = \mathcal{S}_{\Delta d^t}\left(\mathcal{S}_{2d^{t-1}}(w^{t-1} - \eta \nabla_w \mathcal{L}(w^{t-1}))\right), \quad (17)$$

which implies a shrinkage on the length of step followed by a shrinkage on weight.

Case 3: Zeroed weight. For a zeroed weight, the corresponding $\theta \in [-d, d]$. The derivative $\frac{\partial w}{\partial \theta}$ is always zero, which is caused by plateau landscape of mapping $w = \mathcal{S}_d(\theta)$. Hence, w will be fixed to zero after θ occasionally lies in the interval. For this reason, a regrowth mechanism is necessary for reviving pruned weights. Modifying gradient as Grad R (Chen et al., 2021) did is a promising method. Specifically, we manually set $\frac{\partial w}{\partial \theta} \equiv 1$ for zeroed w. Notice that the authors of Grad R proved that under some smooth conditions, the loss will be non-increasing if the learning rate is constant and carefully chosen. Similarly, we adopt the idea and derive a similar explanation for our case of soft threshold mapping as follows

Theorem 4.1 (Convergence)**.** *For a spiking neural network, where each synaptic weight w is dominated by corresponding spine size θ through a soft threshold mapping*

$$w = \text{sign}(\theta) \cdot (|\theta| - d)_+, d \geq 0, \quad (18)$$

if we apply a smooth approximation

$$w = f(\theta) := \frac{1}{\alpha} \log \left(\frac{1 + e^{\alpha(\theta - d)}}{1 + e^{-\alpha(\theta + d)}} \right), \alpha \gg 1, \quad (19)$$

and define the pseudo partial derivative during computing gradients as $\frac{\partial w}{\partial \theta}_p \equiv 1$, the loss function $\mathcal{L}$ is L-smooth and lower bounded, the sequence $\{\mathcal{L}(\boldsymbol{\theta}^t)\}_{t \in \mathbb{N}}$ must converge if learning rate $\eta < \frac{4}{L(1+e^{\alpha d})}$.

The detailed proof is provided in the Appendix B. Distinct from the aforementioned two cases, the step size of zeroed weights is dictated by filopodia size θ and gradient w.r.t. zeroed weight $\frac{\partial \mathcal{L}}{\partial w}\big|_{w=0}$ as opposed to weights themselves. The update rule takes the form

$$\boldsymbol{w}^t = \mathcal{S}_{\Delta d^t}(\mathcal{S}_{d^{t-1}}(\boldsymbol{\theta}^{t-1} - \eta \nabla_{\boldsymbol{w}} \mathcal{L})) \quad (20)$$

Analysis In all the cases of the update rule, the weights are always pushed to zero by a soft shrinkage $\mathcal{S}_{\Delta d^t}(\cdot)$ brought by increasing threshold throughout the training progress, which acts as a proximal gradient descent under ℓ_1-regularized loss as mentioned in Case 1, encouraging the transition from mature spines to filopodia. Despite this, the modified gradient introduced in Case 3 ensures that state transition between different kinds of synapses is unobstructed, theoretically guaranteeing convergence. The nested soft threshold operator in Case 2 suggests it is increasingly difficult to directly flip signs of weight immediately in the late stages of training since d is growing larger. Such a phenomenon indicates that most updates of nonzero weight in the late stages will follow Case 1.

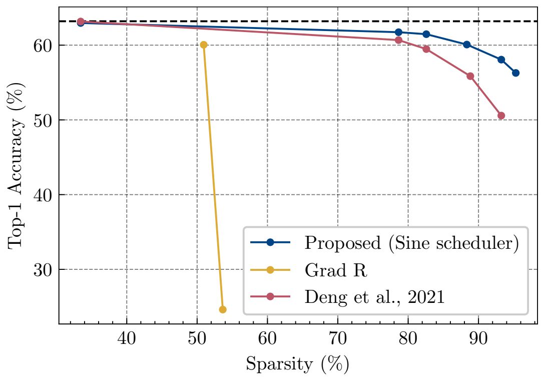

Figure 3. Performance comparison of different pruning algorithms on the ImageNet dataset. The accuracy of unpruned model is shown as a black dashed line.

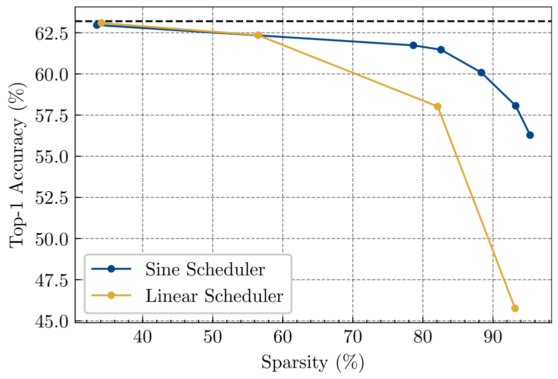

Figure 4. Performance comparison of two schedulers on the ImageNet dataset. The accuracy of unpruned model is shown as a black dashed line.

4.4. Scheduling of Threshold Growth

Deciding on an appropriate schedule of threshold d at each training steps is non-trivial. Assuming the total steps of parameter updating is T_{param}, and the target threshold after training is $D > 0$, which are decided before training, we propose two schedulers:

- Linear scheduler: $d^t = \frac{t}{T_{\text{param}}} D$
- Sine scheduler: $d^t = \frac{1}{2}\left(\sin\left(\frac{t\pi}{T_{\text{param}}} - \frac{\pi}{2}\right) + 1\right) D$

These schedulers show different behaviors and will be further discussed in the experimental results section.

5. Experiments

In this section, we validate the effectiveness of our pruning algorithm for classification tasks on the ImageNet

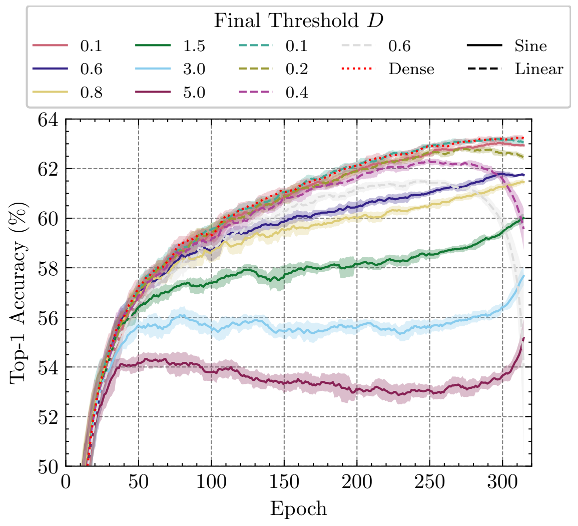

Figure 5. Top-1 accuracy during training (320 epochs in total). Lines are smoothed by window size 10, and the error bars (areas with lighter colors) indicate the standard deviations within the windows. The lines in the early stage are clipped for highlighting the latter stage.

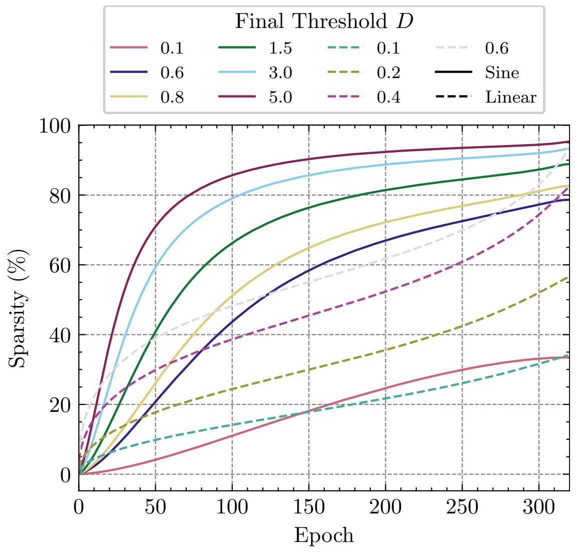

Figure 6. Overall sparsity of parameters in Conv. layers during training (320 epochs in total).

dataset (Russakovsky et al., 2015). We first compare our algorithm with Grad R to demonstrate the significance of transition between positive and negative weights (different types of synapses). Further, we investigate the performance of the two proposed schedulers of threshold d and make an empirical explanation of their different performance. Finally, we compare our method with other state-of-the-art approaches. Considering the training cost of deep SNNs, we select SEW ResNet18 (Fang et al., 2021a) as a testbed and follow ALL the training setting of SEW ResNet. Notice that the computation concerning spikes is from the layer behind neuron layers, which are usually convolutional or fully-connected layers. Adding that pruning the final fully-connected layer will hurt the performance, our pruning algorithm only applies to all convolutional layers. The detailed extra settings for pruning are shown in Appendix A.

5.1. Effects of Transition between Excitatory and Inhibitory Synapses

We implement Grad R (Chen et al., 2021) as a baseline, which also can be viewed as a pruning and regrowth algorithm. In fact, Grad R is a degenerate case of our algorithm, where d is extremely large and fixed during training so that the transition of different types of synapses is forbidden. The comparison of performance with SEW ResNet18 structure on ImageNet dataset is illustrated in Fig. 3. One can find that the performance of our method (blue curve) is much better than Grad R (yellow curve). The performance of Grad R drops drastically even under low sparsity ($\sim$50%) in default of the transition of different types of synapses. It indicates that Grad R fails to achieve impressive performance loss as it did on smaller datasets, which is no match for our proposed algorithm. All these results demonstrate the effects of transition between excitatory and inhibitory synapses.

5.2. Comparison of Threshold Scheduler

As it can be seen, our algorithm has only one tunable hyper-parameter D, which actually transfers the complexity of the parameter space to the function space, namely the choice of schedulers. Here, we compare the performance and other properties of the two proposed schedulers, namely linear scheduler and sine scheduler. As shown in Fig. 4 and Tab. 2, the performance of the linear scheduler drops drastically when the sparsity is above 80%. In contrast, the performance of the sine scheduler is close to the performance of the fully connected network when the connectivity is above 10%. Besides, it reaches a top-1 accuracy of 58.06% ($\sim$5.2% accuracy loss) when constrained to 6.74% connectivity. These results imply that the sine scheduler is superior than the linear scheduler.

To dive deep into the root cause of such performance variation, we further analyze the top-1 accuracy and total sparsity of SNN during training with these two schedulers. Fig. 5 illustrates the top-1 accuracy during training. An interesting finding is that some training processes ($D = 0.4, 0.6$) using the linear scheduler severe performance degradation in the last 50 epochs (270$\sim$320). On the contrary, all processes applying the sine scheduler maintain the rising trend

Table 1. Comparison of performance in recent studies on SNN pruning.

Pruning Method	Training Method	Arch.	Dataset	Top-1 Acc. (%)	Sparsity (%)	Acc. Loss (%)
Martinelli et al., 2020	Surrogate Gradient	2 FC	QUT-NOISE-TIMIT	-4.6[1] -12.4 -25.2	85	-0.1 -0.1 -0.6
Chen et al., 2021	Surrogate Gradient	6 Conv, 2 FC	CIFAR-10	92.84	97.65 99.27	-1.47 -3.52
Deng et al., 2021	Surrogate Gradient	7 Conv, 2 FC	CIFAR-10	89.53	75	-2.15
Kundu et al., 2021	ANN2SNN & Surrogate Gradient	ResNet12 VGG16	CIFAR-100 Tiny-ImageNet	63.52 57.00	90 60	-0.5 -4.3
Nguyen et al., 2021	STDP	3 Conv	Caltech-101 (2 Classes)	95.7	92.83	0
This work	Surrogate Gradient	**6 Conv, 2 FC**	**CIFAR-10**	**92.84**	**97.77** **99.25**	**-0.35** **-2.63**
Chen et al., 2021[2]	Surrogate Gradient	SEW ResNet18	ImageNet	63.22	50.94 53.65	-3.17 -38.6
Deng et al., 2021[2]	Surrogate Gradient	SEW ResNet18	ImageNet	63.22	82.58 88.84 93.24	-3.74 -7.37 -12.65
This work	Surrogate Gradient	**SEW ResNet18**	**ImageNet**	**63.22**	**82.58** **88.84** **93.24** **95.30**	**-1.92** **-3.29** **-5.16** **-6.94**

[1] The accuracy here represents -*HTER* for low, medium and high noise levels, where *HTER*=0.5*MR*+0.5*FAR* denotes the Half Total Error Rate. MR and FAR are miss and false alarm rates
[2] Our implementation.

throughout training. Some trials with the sine schedulers ($D = 1.5, 3.0$) even show an apparent accelerated, increasing accuracy at the final stages. Fig. 6 indicates that these two schedulers show different behavior in the growth of sparsity. The sine scheduler tends to produce a more steady growth of sparsity in the late stages of training, while the linear one usually has a sharp growth.

It should be noted that the scheduler of the learning rate applied in the SEW ResNet is the cosine annealing scheduler (Loshchilov & Hutter, 2017). For the n-th epoch, the learning rate η^n decays as

$$\eta^n = \frac{1}{2}\eta_{\max}\left(1+\cos\left(\frac{n}{T_{\text{epoch}}}\pi\right)\right), \tag{21}$$

where T_{epoch} denotes the total number of training epochs. In the last few epochs, the learning rate is quite small, suggesting that the update step should be infinitesimal, which matches the Case 1 in Section 4.3. Recall that Eq. (15) shows the equivalent optimization problem under ℓ_1-regularized loss. In the context of the linear scheduler, the regularizing term $\Delta d^t \|\boldsymbol{w}\|_1 \equiv \frac{1}{T_{\text{param}}} D\|\boldsymbol{w}\|_1$ is invariant and far outweigh the update step at last, resulting in a pure shrinkage of weights in the absence of learning signal. For the sine scheduler, since $T_{\text{param}} \gg 1$, we have the following approximation

$$\begin{aligned}\Delta d^t &= \frac{1}{2}\left(\sin\left(\frac{t\pi}{T_{\text{param}}}-\frac{\pi}{2}\right)-\sin\left(\frac{(t-1)\pi}{T_{\text{param}}}-\frac{\pi}{2}\right)\right)D\\ &= \sin\left(\frac{(t-\frac{1}{2})\pi}{T_{\text{param}}}\right)\sin\left(\frac{\pi}{T_{\text{param}}}\right)D\\ &\approx \sin\left(\frac{t}{T_{\text{param}}}\pi\right)\frac{\pi}{T_{\text{param}}}D\end{aligned} \tag{22}$$

which converges to zero when $t \to T_{\text{param}}$. In particular, d converges faster than η for $T_{\text{param}} \gg T_{\text{epoch}}$, ensuring the magnitude of the learning signal is always over the regularized term and thereby facilitating learning until the last moment.

5.3. Comparison with the State-of-the-Art

We compare the proposed method to some other state-of-the-art pruning algorithms of SNNs and report the results in Tab. 1. To make a fair comparison, we also conduct experiments on shallow networks and small-scale datasets like CIFAR-10. The results are shown in Fig. 7.

For the trials on CIFAR-10, we follow the setting in Grad R that pruning is applied to all layers except batch normal-

Table 2. Comparison of different threshold schedulers on the ImageNet dataset.

Scheduler	Top-1 Acc. (Dense) (%)	Sparsity (%)	Top-1 Acc. (Pruned) (%)	Acc. Loss (%)
Sine	63.22	33.45	62.84	-0.38
		78.64	61.51	-1.71
		82.58	61.30	-1.92
		88.84	59.93	-3.29
		93.24	58.06	-5.16
		95.30	56.28	-6.94
Linear		34.10	62.99	-0.23
		56.49	62.28	-0.94
		82.11	58.02	-5.20
		93.15	45.76	-17.46

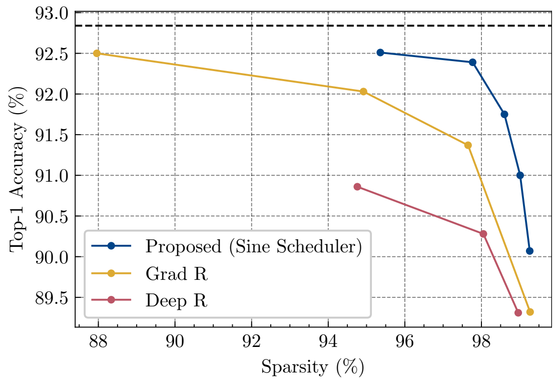

Figure 7. Performance comparison of different pruning algorithms on the CIFAR-10 dataset. The accuracy of unpruned model is shown as a black dashed line.

ization layers. The detailed setting of hyperparameters in trials on CIFAR-10 are shown in Appendix A. Most of these works fail to achieve the low performance loss as ours even on datasets much simpler than ImageNet, which demonstrates the advanced nature of our approach. Besides, we can find that Grad R, which has an impressive performance on CIFAR-10, cannot generalize to larger datasets and networks.

6. Conclusion and Discussion

In this paper, we design a novel pruning algorithm based on modeling the transition of dendritic spines containing different types of neurotransmitters and the transition between development stages of dendritic spines. We theoretically explain the convergence and the source of the sparsity in our approach. Extensive experimental results show that our approach outperforms previous SNN pruning algorithms by a substantial degree, which encourages the subsequent work to validate its performance on more realistic tasks. The results also reveal the significance of choice in threshold scheduler, which is correlated to the scheduler of learning rates. The optimal threshold scheduler for our pruning algorithm remains to be discussed and discovered in future work.

Acknowledgements

This work is supported by grants from the National Natural Science Foundation of China under contract No. 62027804, No. 61825101, No. 62088102 and No. 62176003. The computing resources of Pengcheng Cloudbrain are used in this research.

References

Bé̈ïque, J.-C., Lin, D.-T., Kang, M.-G., Aizawa, H., Takamiya, K., and Huganir, R. L. Synapse-specific regulation of ampa receptor function by psd-95. *Proceedings of the National Academy of Sciences*, 103(51):19535–19540, 2006.

Bellec, G., Kappel, D., Maass, W., and Legenstein, R. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018a.

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., and Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018b.

Bhat, R., Axtell, R., Mitra, A., Miranda, M., Lock, C., Tsien, R. W., and Steinman, L. Inhibitory role for gaba in autoimmune inflammation. *Proceedings of the National Academy of Sciences*, 107(6):2580–2585, 2010.

Chechik, G., Meilijson, I., and Ruppin, E. Synaptic pruning in development: A computational account. *Neural Computation*, 10(7):1759–1777, 1998.

Chechik, G., Meilijson, I., and Ruppin, E. Neuronal Regulation: A Mechanism for Synaptic Pruning During Brain Maturation. *Neural Computation*, 11(8):2061–2080, 11 1999.

Chen, Y., Yu, Z., Fang, W., Huang, T., and Tian, Y. Pruning of deep spiking neural networks through gradient rewiring. In Zhou, Z.-H. (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1713–1721. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

Daubechies, I., Defrise, M., and De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

Davies, M., Srinivasa, N., Lin, T., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., Weng, Y., Wild, A., Yang, Y., and Wang, H. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

Deng, L., Wu, Y., Hu, Y., Liang, L., Li, G., Hu, X., Ding, Y., Li, P., and Xie, Y. Comprehensive snn compression using admm optimization and activity regularization. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

Engert, F. and Bonhoeffer, T. Dendritic spine changes associated with hippocampal long-term synaptic plasticity. *Nature*, 399(6731):66–70, 1999.

Fang, W., Chen, Y., Ding, J., Chen, D., Yu, Z., Zhou, H., Tian, Y., and other contributors. Spikingjelly. `https://github.com/fangwei123456/spikingjelly`, 2020. Accessed: 2022-01-18.

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., and Tian, Y. Deep residual learning in spiking neural networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 21056–21069. Curran Associates, Inc., 2021a.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2661–2671, October 2021b.

Fiala, J. C., Feinberg, M., Popov, V., and Harris, K. M. Synaptogenesis via dendritic filopodia in developing hippocampal area ca1. *Journal of Neuroscience*, 18(21): 8900–8911, 1998.

Fonnum, F. Glutamate: a neurotransmitter in mammalian brain. *Journal of neurochemistry*, 42(1):1–11, 1984.

Furber, S. B., Galluppi, F., Temple, S., and Plana, L. A. The spinnaker project. *Proceedings of the IEEE*, 102(5): 652–665, 2014.

Gerstner, W. and Kistler, W. M. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Hu, Y., Tang, H., and Pan, G. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–6, 2021.

Kappel, D., Habenschuss, S., Legenstein, R., and Maass, W. Network plasticity as bayesian inference. *PLOS Computational Biology*, 11(11):1–31, 11 2015.

Kheradpisheh, S. R., Mirsadeghi, M., and Masquelier, T. Bs4nn: Binarized spiking neural networks with temporal coding and learning. *Neural Processing Letters*, pp. 1–19, 2021.

Kim, S., Park, S., Na, B., and Yoon, S. Spiking-yolo: Spiking neural network for energy-efficient object detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11270–11277, Apr. 2020.

Kirkland, P., Di Caterina, G., Soraghan, J., and Matich, G. Spikeseg: Spiking segmentation via stdp saliency mapping. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020.

Kundu, S., Datta, G., Pedram, M., and Beerel, P. A. 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 (WACV)*, pp. 3953–3962, January 2021.

Kushawaha, R. K., Kumar, S., Banerjee, B., and Velmurugan, R. Distilling spikes: Knowledge distillation in spiking neural networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 4536–4543, 2021.

Kusupati, A., Ramanujan, V., Somani, R., Wortsman, M., Jain, P., Kakade, S., and Farhadi, A. Soft threshold weight reparameterization for learnable sparsity. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5544–5555. PMLR, 13–18 Jul 2020.

Liu, Y., Qian, K., Hu, S., An, K., Xu, S., Zhan, X., Wang, J. J., Guo, R., Wu, Y., Chen, T.-P., Yu, Q., and Liu, Y. Application of deep compression technique in spiking neural network chip. *IEEE Transactions on Biomedical Circuits and Systems*, 14(2):274–282, 2020.

Loshchilov, I. and Hutter, F. SGDR: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Luo, Y., Yi, Q., Wang, T., Lin, L., Xu, Y., Zhou, J., Yuan, C., Guo, J., Feng, P., and Feng, Q. A spiking neural network architecture for object tracking. In Zhao, Y., Barnes, N., Chen, B., Westermann, R., Kong, X., and Lin, C. (eds.), *Image and Graphics*, pp. 118–132, Cham, 2019. Springer International Publishing.

Luo, Y., Xu, M., Yuan, C., Cao, X., Zhang, L., Xu, Y., Wang, T., and Feng, Q. Siamsnn: Siamese spiking neural networks for energy-efficient object tracking. In Farkaš, I., Masulli, P., Otte, S., and Wermter, S. (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2021*, pp. 182–194, Cham, 2021. Springer International Publishing.

Ma, D., Shen, J., Gu, Z., Zhang, M., Zhu, X., Xu, X., Xu, Q., Shen, Y., and Pan, G. Darwin: A neuromorphic hardware co-processor based on spiking neural networks. *Journal of Systems Architecture*, 77:43 – 51, 2017.

Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9): 1659–1671, 1997.

Martinelli, F., Dellaferrera, G., Mainar, P., and 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, 2020.

Matsuzaki, M., Ellis-Davies, G. C. R., Nemoto, T., Miyashita, Y., Iino, M., and Kasai, H. Dendritic spine geometry is critical for ampa receptor expression in hippocampal ca1 pyramidal neurons. *Nature Neuroscience*, 4(11):1086–1092, 2001.

Matsuzaki, M., Honkura, N., Ellis-Davies, G. C. R., and Kasai, H. Structural basis of long-term potentiation in single dendritic spines. *Nature*, 429(6993):761–766, 2004.

Mern, J., Gupta, J. K., and Kochenderfer, M. J. Layer-wise synapse optimization for implementing neural networks on general neuromorphic architectures. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2017.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., Brezzo, B., Vo, I., Esser, S. K., Appuswamy, R., Taba, B., Amir, A., Flickner, M. D., Risk, W. P., Manohar, R., and Modha, D. S. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197): 668–673, 2014.

Mink, J. W., Blumenschine, R. J., and Adams, D. B. Ratio of central nervous system to body metabolism in vertebrates: its constancy and functional basis. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 241(3):R203–R212, 1981.

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

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

Neil, D., Pfeiffer, M., and Liu, S.-C. Learning to be efficient: Algorithms for training low-latency, low-compute deep spiking neural networks. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pp. 293–298, New York, NY, USA, 2016. Association for Computing Machinery.

Nesterov, Y. Gradient methods for minimizing composite objective function (technical report 2007/76). *CORE, Université catholique de Louvain*, 2007.

Nguyen, T. N. N., Veeravalli, B., and Fong, X. Connection pruning for deep spiking neural networks with on-chip learning. In *International Conference on Neuromorphic Systems 2021*, ICONS 2021, New York, NY, USA, 2021. Association for Computing Machinery.

Noguchi, J., Matsuzaki, M., Ellis-Davies, G. C., and Kasai, H. Spine-neck geometry determines nmda receptor-dependent ca2+ signaling in dendrites. *Neuron*, 46(4): 609–622, 2005.

Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A., and Naud, R. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature Neuroscience*, 24(7):1010–1019, 2021.

Pei, J., Deng, L., Song, S., Zhao, M., Zhang, Y., Wu, S., Wang, G., Zou, Z., Wu, Z., He, W., Chen, F., Deng, N., Wu, S., Wang, Y., Wu, Y., Yang, Z., Ma, C., Li, G., Han, W., Li, H., Wu, H., Zhao, R., Xie, Y., and Shi, L. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

Perez-Nieves, N. and Goodman, D. Sparse spiking gradient descent. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 11795–11808. Curran Associates, Inc., 2021.

Qiao, G., Ning, N., Zuo, Y., Hu, S., Yu, Q., and Liu, Y. Direct training of hardware-friendly weight binarized spiking neural network with surrogate gradient learning towards spatio-temporal event-based dynamic data recognition. *Neurocomputing*, 457:203–213, 2021.

Rathi, N., Panda, P., and 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, 2019.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Stöckl, C. and Maass, W. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Machine Intelligence*, 3(3): 230–238, 2021.

Stromatias, E., Neil, D., Pfeiffer, M., Galluppi, F., Furber, S. B., and Liu, S.-C. Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms. *Frontiers in Neuroscience*, 9:222, 2015.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Takuya, S., Zhang, R., and Nakashima, Y. Training low-latency spiking neural network through knowledge distillation. In *2021 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*, pp. 1–3, 2021.

Tønnesen, J., Katona, G., Rózsa, B., and Nägerl, U. V. Spine neck plasticity regulates compartmentalization of synapses. *Nature neuroscience*, 17(5):678–685, 2014.

Wang, Y., Xu, Y., Yan, R., and Tang, H. Deep spiking neural networks with binary weights for object recognition. *IEEE Transactions on Cognitive and Developmental Systems*, 13(3):514–523, 2021.

Wu, J., Xu, C., Han, X., Zhou, D., Zhang, M., Li, H., and Tan, K. C. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12: 331, 2018.

Yin, H., Lee, J. B., Kong, X., Hartvigsen, T., and Xie, S. Energy-efficient models for high-dimensional spike train classification using sparse spiking neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2017–2025, 2021.

Yuste, R. *Dendritic spines*. MIT press, 2010.

Zenke, F. and Ganguli, S. SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 30(6):1514–1541, 06 2018.

Zenke, F. and Neftci, E. O. Brain-inspired learning on neuromorphic substrates. *Proceedings of the IEEE*, 109 (5):935–950, 2021.

Zenke, F. and Vogels, T. P. The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks. *Neural Computation*, 33(4):899–925, 03 2021.

Zheng, H., Wu, Y., Deng, L., Hu, Y., and Li, G. Going deeper with directly-trained larger spiking neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11062–11070, May 2021.

Zhou, S., Li, X., Chen, Y., Chandrasekaran, S. T., and Sanyal, A. Temporal-coded deep spiking neural network with easy training and robust performance. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12): 11143–11151, May 2021.

Zuo, Y., Lin, A., Chang, P., and Gan, W.-B. Development of long-term dendritic spine stability in diverse regions of cerebral cortex. *Neuron*, 46(2):181–189, 2005.

A. Detailed Setting of Pruning Experiments

A.1. Setting of Our Propose Method

There is only one hyperparameter in our algorithm, namely D, which represents the final width of the “plateau” landscape as illustrated in Fig. 1. The D in our trials and corresponding performances are shown in Tab. 3 and Tab. 4.

Table 3. Experimental settings of proposed algorithm on ImageNet.

Scheduler	D	Sparsity (%)	Top-1 Acc. (Pruned) (%)	Acc. Loss (%)
Sine	0.1	33.45	62.84	-0.38
	0.6	78.64	61.51	-1.71
	0.8	82.58	61.30	-1.92
	1.5	88.84	59.93	-3.29
	3.0	93.24	58.06	-5.16
	5.0	95.30	56.28	-6.94
Linear	0.1	34.10	62.99	-0.23
	0.2	56.49	62.28	-0.94
	0.4	82.11	58.02	-5.20
	0.6	93.15	45.76	-17.46

Table 4. Experimental settings of proposed algorithm on CIFAR-10.

Scheduler	D	Sparsity (%)	Top-1 Acc. (Pruned) (%)	Acc. Loss (%)
Sine	1.0	95.36	92.49	-0.35
	2.0	97.77	92.49	-0.35
	3.0	98.6	91.64	-1.20
	4.0	99.01	91.06	-1.78
	5.0	99.25	90.21	-2.63

All the convolutional layer in SEW ResNet18 are pruned.

A.2. Setting of Grad R

For the reproduction of the Grad R algorithm (Chen et al., 2021), there are several parameters including scale and location parameter α, μ, which are related to the target sparsity p as

$$\mu\alpha = \log(2 - 2p) \tag{23}$$

Our experimental settings are shown in Tab. 5.

Table 5. Experimental settings of our implementation of Grad R on ImageNet.

μ	α	p	Sparsity (%)	Top-1 Acc. (Pruned) (%)	Acc. Loss (%)
-100	0	0.5	50.94	60.052	-3.168
-100	10^{-5}	0.5005	53.65	24.616	-38.604

A.3. Setting of (Deng et al., 2021)

For the reproduction of (Deng et al., 2021), most retraining hyperparameters are inherited from training of dense model, which has a 63.22% top-1 accuracy. The extra hyperparameters are shown in Tab. 6.

The sparsity of each pruned layer are set identical to the corresponding sparse model using sine scheduler.

Table 6. Experimental settings of our implementation of (Deng et al., 2021) on ImageNet.

Parameters	Descriptions	Value
N_1	Retraining Epochs for ADMM-Compression	40
N_2	Retraining Epochs for Hard-Compression(HC)	20
ρ	Penalty Coefficient for ADMM	1e-4
lr	Initial Learning Rate during ADMM and HC Retraining	0.05
-	Optimizer during ADMM and HC Retraining	SGD
-	ℓ_2 Penalty	5e-5
-	Learning Rate Scheduler	Cosine Annealing

A.4. Software and Hardware Platform

The implementation is based on the SpikingJelly framework (Fang et al., 2020). Each trial on ImageNet is carried out on 8 NVIDIA V100 GPUs. Each trial on CIFAR-10 is carried out on a single NVIDIA V100 GPU.

B. Proof of Theorem 4.1

Theorem B.1 (Convergence)**.** *For a spiking neural network, where each synaptic weight w is dominated by corresponding spine size θ through a soft threshold mapping*

$$w = \operatorname{sign}(\theta) \cdot (|\theta| - d)_+, d \geq 0, \tag{24}$$

if we apply a smooth approximation

$$w = f(\theta) := \frac{1}{\alpha} \log \left(\frac{1 + e^{\alpha(\theta - d)}}{1 + e^{-\alpha(\theta + d)}} \right), \alpha \gg 1, \tag{25}$$

and define the pseudo partial derivative during computing gradients as $\frac{\partial w}{\partial \theta}_p \equiv 1$, the loss function $\mathcal{L}$ is L-smooth and lower bounded, the sequence $\{\mathcal{L}(\boldsymbol{\theta}^t)\}_{t\in\mathbb{N}}$ must converge if learning rate $\eta < \frac{4}{L(1+e^{\alpha d})}$.

Proof. Denote the pseudo gradient calculated using $\frac{\partial w}{\partial \theta}_p \equiv 1$ as $\nabla_p \mathcal{L}(\boldsymbol{\theta})$, the update follows the rule $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta \nabla_p \mathcal{L}(\boldsymbol{\theta})$. Since $\mathcal{L}$ is L-smooth , we have

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}^{t+1}) - \mathcal{L}(\boldsymbol{\theta}^t) &\leq \langle \nabla \mathcal{L}(\boldsymbol{\theta}^t), \boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t \rangle + \frac{L}{2} \|\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t\|^2 \\
&= -\eta \langle \nabla \mathcal{L}(\boldsymbol{\theta}^t), \nabla_p \mathcal{L}(\boldsymbol{\theta}^t) \rangle + \frac{L\eta^2}{2} \|\nabla_p \mathcal{L}(\boldsymbol{\theta}^t)\|^2
\end{aligned} \tag{26}$$

Notice that

$$\frac{\|\nabla_p \mathcal{L}(\boldsymbol{\theta})\|^2}{\langle \nabla \mathcal{L}(\boldsymbol{\theta}), \nabla_p \mathcal{L}(\boldsymbol{\theta}) \rangle} = \frac{\sum_k \left(\frac{\partial \mathcal{L}}{\partial w_k} \right)^2}{\sum_k \left[\frac{\partial w_k}{\partial \theta_k} \left(\frac{\partial \mathcal{L}}{\partial w_k} \right)^2 \right]} \tag{27}$$

and the actual derivative has the form

$$\frac{\partial w}{\partial \theta} = f'(\theta) = \frac{1 + e^{2\alpha\theta} + 2e^{\alpha(\theta - d)}}{(1 + e^{\alpha(\theta - d)})(1 + e^{\alpha(\theta + d)})}. \tag{28}$$

It is clear that $f'(\theta)$ is continuously differentiable on $\mathbb{R}$ and

$$f''(\theta) = \frac{4\alpha e^{2\alpha\theta} \sinh(\alpha d) \sinh(\alpha\theta)}{\left[(1 + e^{\alpha(\theta - d)})(1 + e^{\alpha(\theta + d)}) \right]^2} \tag{29}$$

has a unique zero point $\theta = 0$. It is actually a global minimum $f'(0) = \frac{2}{1+e^{\alpha d}}$ and thereby we have

$$\begin{aligned}\frac{\|\nabla_p \mathcal{L}(\boldsymbol{\theta})\|^2}{\langle \nabla \mathcal{L}(\boldsymbol{\theta}), \nabla_p \mathcal{L}(\boldsymbol{\theta}) \rangle} &\leq \frac{\sum_k \left(\frac{\partial \mathcal{L}}{\partial w_k} \right)^2}{\sum_k \left[f'(0) \left(\frac{\partial \mathcal{L}}{\partial w_k} \right)^2 \right]} \\ &= \frac{1+e^{\alpha d}}{2}\end{aligned} \tag{30}$$

Denote $A = \frac{1+e^{\alpha d}}{2}$, the last term in Eq. (26) can be further estimate as

$$\mathcal{L}(\boldsymbol{\theta}^{t+1}) - \mathcal{L}(\boldsymbol{\theta}^t) \leq -\eta (1 - \frac{AL\eta}{2}) \langle \nabla \mathcal{L}(\boldsymbol{\theta}^t), \nabla_p \mathcal{L}(\boldsymbol{\theta}^t) \rangle \tag{31}$$

Since $\langle \nabla \mathcal{L}(\boldsymbol{\theta}^t), \nabla_p \mathcal{L}(\boldsymbol{\theta}^t) \rangle \geq A \|\nabla_p \mathcal{L}(\boldsymbol{\theta}^t)\|^2 \geq 0$, to form a non-increasing sequence $\{\mathcal{L}(\boldsymbol{\theta}^t)\}_{t \in \mathbb{N}}$, a proper learning rate should satisfy $\eta < 2/AL = \frac{4}{L(1+e^{\alpha d})}$.

Under the above conditions, the lower bounded, non-increasing loss sequence must converge, which completes the proof. Further, the best choice is $\eta = 1/AL = \frac{2}{L(1+e^{\alpha d})}$. $\square$