POST: Continuous Probabilistic Optimization for Structured Dynamic Sparse Training

Zequan Wang, Bin Wang, Haodong Bian, Xiaochun Yang, Man Yuan

School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
2472057@stu.neu.edu.cn, binwang@mail.neu.edu.cn, 2290168@stu.neu.edu.cn,
yangxc@mail.neu.edu.cn, 2401953@stu.neu.edu.cn

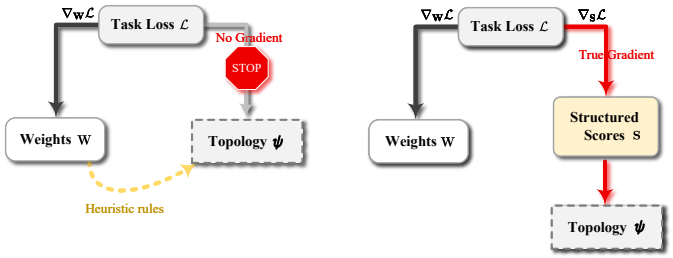
***Abstract*—Dynamic Sparse Training (DST) serves as a promising paradigm for training efficient deep neural networks from scratch by dynamically adjusting the sparse topology. Although unstructured DST methods can achieve high sparsity, they fail to realize practical acceleration on general-purpose hardware. Conversely, while existing structured DST methods can improve inference speed, they typically rely on sub-optimal heuristic masking criteria or rigid coarse-grained constraints, leading to poor convergence and limited model expressivity. To address these challenges, we propose POST, a novel framework that reformulates the discrete mask search into a continuous probabilistic optimization problem. By introducing learnable structural scores, we establish a gradient pathway directly from the task loss to network topology. This enables the network to leverage topological gradients for structure learning, effectively resolving the misalignment between heuristic indicators and the actual training objective. Furthermore, to balance flexibility with hardware efficiency, we adopt a hybrid topology evolution strategy that combines fine-grained Constant Fan-in constraints with importance-aware neuron pruning, enabling practical acceleration via N:M sparsity. Experiments on CIFAR-10/100 and ImageNet-1K demonstrate that POST outperforms existing structured DST approaches in accuracy.**

***Index Terms*—Dynamic Sparse Training, Structured Pruning, Model Compression**

I. Introduction

Deep Neural Networks (DNNs) have achieved unprecedented success in the field of artificial intelligence; however, their ever-growing complexity incurs prohibitive computational and memory costs [1], [2]. This bottleneck severely restricts the deployment of the model on resource-constrained edge devices. To overcome this limitation, Dynamic Sparse Training (DST) has emerged as a highly competitive alternative. It enables dynamic adjustment of the connection topology throughout the optimization process, allowing for the training of sparse networks directly from scratch. This paradigm significantly reduces the training resource budget and facilitates the exploration of a broader range of architectural possibilities. While current DST methods [3], [4] can match the performance of dense models at 80%–95% sparsity, they typically yield unstructured sparse patterns, making it difficult to achieve substantial inference acceleration on off-the-shelf general-purpose hardware accelerators (e.g., GPUs).

In contrast, Structured Dynamic Sparse Training methods, such as SRigL [5], have successfully achieved inference acceleration by enforcing N:M sparsity patterns [6]. However,



(a) Structured DST (b) POST

Fig. 1. Illustration of the gradient flow. (a) Traditional methods rely on heuristic rules. (b) Our framework leverages structural score gradients to strictly align topology updates with the loss minimization objective.

as shown in Fig. 1, these approaches inherit the topological update strategy of RigL [3], relying primarily on heuristic rules like weight magnitude and gradients. This reliance on proxy metrics rather than direct loss optimization creates a potential misalignment between structural evolution and the actual training objective, often preventing the algorithm from converging to the optimal subnetwork [7]. On the other hand, methods like DynaDiag [8] leverage custom CUDA kernels to ensure hardware efficiency by strictly enforcing diagonal constraints (specifically, dynamically selectable diagonals). However, this adaptation strategy is inherently coarse-grained, operating at the diagonal level—shifting entire groups of connections simultaneously—rather than allowing for element-wise flexibility. While computationally efficient, this introduces a strong inductive bias that limits the upper bound of the model's expressivity. Therefore, a critical challenge remains: how to maximize model accuracy while maintaining computational efficiency.

To address this challenge, we propose POST (Probabilistic Optimization for Structured Topology), a novel framework that reformulates topology search from discrete heuristic selection into continuous probabilistic optimization. Unlike traditional methods, POST establishes a fully differentiable paradigm to explore optimal structures through two key mechanisms: (1) **Gradient-based Differentiable Structure Learning**. To resolve the misalignment between heuristic indicators like weight magnitude and the actual training objective, we introduce learnable structural scores and establish a direct gradient pathway from the task loss to the topology utilizing Gumbel-

Softmax relaxation. This allows the network to leverage "True Topological Gradients" rather than proxy metrics to guide the search, ensuring that every structural update strictly follows the direction of loss minimization, effectively avoiding sub-optimal solutions. (2) **Hardware-Friendly Hybrid Topology Evolution**. To balance flexibility with hardware efficiency, we propose a strategy combining element-wise constant fan-in constraints (a special case of N:M sparsity) [9] with coarse-grained neuron pruning. The constant fan-in constraint ensures fine-grained connectivity patterns compatible with NVIDIA Sparse Tensor Cores for acceleration, while coarse-grained neuron pruning further eliminates redundant channels to reduce model size. This hybrid design enables the network to leverage hardware acceleration while flexibly allocating the parameter budget to capture critical features.

Our main contributions are summarized as follows:

- We propose the POST framework, which reformulates discrete structured pruning as a continuous differentiable optimization problem. Using structural score gradients, POST ensures that structural updates inherently align with the objective of loss minimization, effectively bypassing the limitations of rigid heuristic rules to discover superior sparse structures.
- We employ a Hardware-Friendly Hybrid Topology Evolution strategy. This approach integrates element-wise constant fan-in constraints with coarse-grained neuron pruning, utilizing an importance-aware dual-metric mechanism to accurately identify and prune redundant neurons. This allows for redirecting the parameter budget to information-rich regions to maximize model expressivity while supporting efficient hardware acceleration.
- We perform extensive evaluations on the CIFAR-10/100 and ImageNet datasets. The results demonstrate that POST outperforms existing structured dynamic sparse training methods in terms of accuracy, achieving a superior trade-off between model performance and computational efficiency.

II. Notations and Preliminaries

Consider a neural network layer parameterized by a dense weight matrix $\mathbf{W}$ and receiving an input feature vector $\mathbf{x}$. To represent the sparse connectivity topology, we introduce a binary mask $\boldsymbol{\psi}$ of the same shape as $\mathbf{W}$. The sparse forward propagation is formulated by applying the mask to the weights via element-wise multiplication ($\odot$), followed by the linear transformation and non-linear activation $\sigma(\cdot)$:

$$\mathbf{y} = \sigma\left((\mathbf{W} \odot \boldsymbol{\psi})\mathbf{x}\right). \tag{1}$$

The overall training objective is to jointly optimize the weights and the topology to minimize the task-specific loss function $\mathcal{L}(\cdot)$ over the data distribution $\mathcal{D}$:

$$\min_{\mathbf{W},\boldsymbol{\psi}} \mathbb{E}_{(\mathbf{x},\mathbf{t})\sim\mathcal{D}}\left[\mathcal{L}(f(\mathbf{x};\mathbf{W}\odot\boldsymbol{\psi}),\mathbf{t})\right], \tag{2}$$

where $\mathbf{t}$ represents the target labels. To enable differentiable optimization for the discrete binary mask $\boldsymbol{\psi}$, we introduce a continuous latent variable matrix $\mathbf{S}$, termed *structural scores*, which serves as the learnable parameterization for generating the binary mask.

Dynamic Sparse Training: To reduce computational costs and memory usage, DST trains a randomly initialized sparse neural network $f(\mathbf{x};\mathbf{W}\odot\boldsymbol{\psi})$ from scratch, dynamically adjusting the sparsity topology $\boldsymbol{\psi}$ throughout the training process. The standard DST scheme operates as an iterative process that alternates between parameter optimization and topology evolution. During the parameter optimization phase, the non-zero weights in $\mathbf{W}$ are updated via stochastic gradient descent to minimize the task-specific loss while keeping the topology fixed. Subsequently, in the topology evolution phase, the mask is adjusted to explore superior structures. This evolution typically involves pruning connections deemed unimportant and growing new connections with high potential.

Constant Fan-in Structured Sparsity: To ensure hardware efficiency, we adopt the Constant Fan-in constraint. Unlike unstructured pruning which suffers from irregular memory access, this paradigm mandates a fixed number of active input connections for every output neuron, ensuring strict load balancing across threads. We utilize a Condensed Representation to losslessly store sparse weights and indices in dense formats. The high regularity of this structure effectively avoids thread divergence on GPUs, allowing forward propagation to be executed as efficient parallelized vector operations, thereby translating theoretical FLOPs reduction into tangible wall-clock speedups.

III. Methodology

Traditional structured dynamic sparse training typically encounters two fundamental limitations: the reliance on heuristic metrics, such as weight magnitude and gradients, often results in a misalignment with the true training objective, while the enforcement of rigid coarse-grained constraints restricts the model's representational capacity. To address these challenges, we propose the POST framework, which fundamentally reformulates topology search from a discrete selection process into a fully differentiable probabilistic optimization problem, ensuring that topological updates are strictly driven by the goal of minimizing the task loss. By integrating a gradient-based structure learning mechanism with a hybrid evolution strategy—combining fine-grained constant fan-in constraints with coarse-grained neuron pruning—our approach maximizes model accuracy while guaranteeing hardware efficiency. In this section, we detail the mathematical formulation and implementation of these core components.

A. Differentiable Mask Generation via Structural Scores

In this section, we detail the specifics of the differentiable masking framework, where we parameterize the network topology using learnable structural scores. To enable efficient exploration of the topological space, we assign a real-valued structural score matrix $\mathbf{S}$ to the weights and introduce stochasticity via the Gumbel-Max trick [10]. Specifically, we first draw independent and identically distributed samples

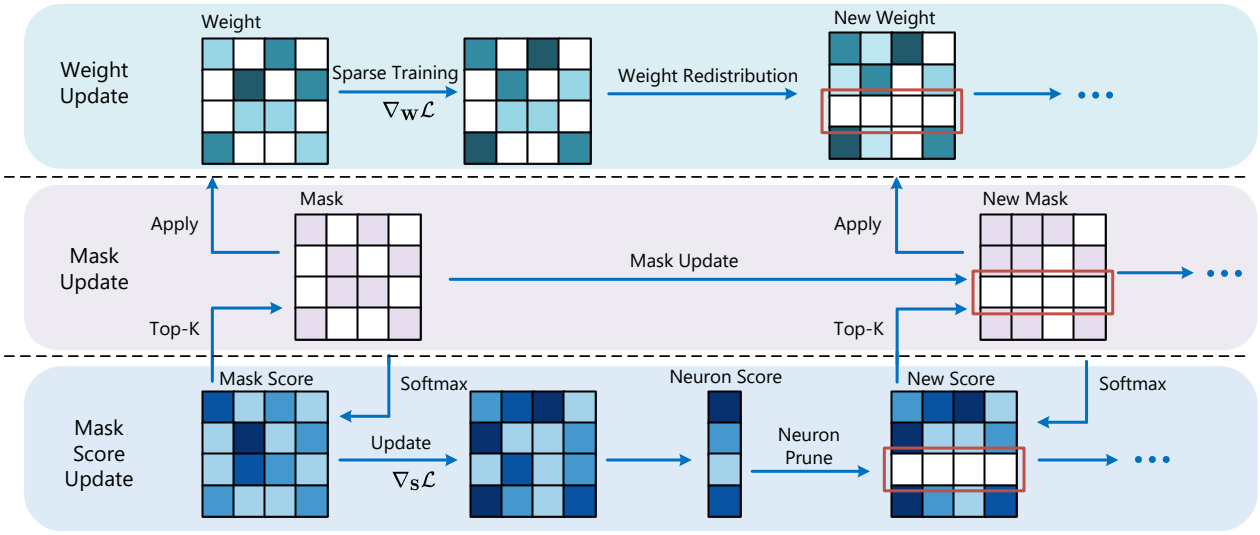


Fig. 2. Schematic illustration of the POST algorithm flow. The framework establishes a continuous evolutionary cycle: sparse weights and structural scores are jointly optimized to update the dynamic mask. Subsequently, the topology evolves by pruning redundant neurons based on importance scores, followed by Weight Redistribution to reallocate connections to the refined architecture.

$\mathbf{G} = -\log(-\log(\mathbf{U}))$ from the standard Gumbel distribution $\text{Gumbel}(0,1)$, where $\mathbf{U} \sim \text{Uniform}(0,1)$ represents the uniform noise matrix. These samples are added to the structural scores to obtain the perturbed scores $\tilde{\mathbf{S}}$:

$$\tilde{\mathbf{S}} = \mathbf{S} + \mathbf{G}, \tag{3}$$

To strictly enforce the constant fan-in constraint, we perform a top-K selection on $\tilde{\mathbf{S}}$ along the fan-in dimension. Based on the selected indices, we generate the discrete binary mask $\boldsymbol{\psi}$ for forward propagation by setting the corresponding connections to 1 and zeroing out the others.

As shown in the mask score update module of Fig. 2, addressing the non-differentiability of the discrete top-K sampling, we employ a continuous relaxation strategy during the backward pass. We approximate the discrete binary mask $\boldsymbol{\psi}$ using a soft mask $\hat{\boldsymbol{\psi}}$, generated by restricting the Gumbel-Softmax distribution to the selected support set with a temperature τ:

$$\hat{\boldsymbol{\psi}} = \text{Softmax}(\tilde{\mathbf{S}}/\tau), \tag{4}$$

where the gradients for unselected connections are approximated as 0. Using this relaxation, we explicitly derive the gradient of the loss $\mathcal{L}$ with respect to the structural score matrix $\mathbf{S}$ via the chain rule:

$$\nabla_{\mathbf{S}}\mathcal{L} \approx \nabla_{\boldsymbol{\psi}}\mathcal{L}\frac{\partial\hat{\boldsymbol{\psi}}}{\partial\tilde{\mathbf{S}}}. \tag{5}$$

Here, $\frac{\partial\hat{\boldsymbol{\psi}}}{\partial\tilde{\mathbf{S}}}$ represents the Jacobian matrix of the Softmax function, which redistributes gradients based on the relative confidence of the scores. Finally, the structural scores are iteratively updated using the computed gradients with a learning rate η:

$$\mathbf{S} \leftarrow \mathbf{S} - \eta\nabla_{\mathbf{S}}\mathcal{L}. \tag{6}$$

This mechanism encourages the model to automatically identify and reinforce connections that contribute most to loss reduction, thereby realizing the dynamic evolution of the topological structure.

B. Gradient Reuse for Efficient Co-Optimization

Optimizing auxiliary structural parameters often incurs significant computational overhead. We mitigate this by leveraging the Shared Activation Gradient mechanism, which couples the learning of topology and weights by recycling the error signal.

Let $\mathcal{L}$ denote the global objective function. During backpropagation, the error signal propagates to the current layer's pre-activation output vector $\mathbf{z}$, yielding the activation gradient $\boldsymbol{\delta} = \nabla_{\mathbf{z}}\mathcal{L}$. This term $\boldsymbol{\delta}$ encapsulates the global error information back-propagated from subsequent layers and represents the most computationally expensive component of the backward pass. In our framework, we reuse this single signal to simultaneously derive the update directions for both the weights $\mathbf{W}$ and the structural scores $\mathbf{S}$.

Applying the chain rule, the gradients for the weights and structural scores are formulated using matrix notation as:

$$\nabla_{\mathbf{W}}\mathcal{L} = (\boldsymbol{\delta}\mathbf{x}^\top) \odot \boldsymbol{\psi}, \tag{7}$$

$$\nabla_{\mathbf{S}}\mathcal{L} \approx (\boldsymbol{\delta}\mathbf{x}^\top) \odot \mathbf{W} \odot \frac{\partial\hat{\boldsymbol{\psi}}}{\partial\tilde{\mathbf{S}}}. \tag{8}$$

As shown in (7) and (8), both gradients rely on the common dense gradient matrix $(\boldsymbol{\delta}\mathbf{x}^\top)$. Consequently, the expensive backpropagation process to compute $\boldsymbol{\delta}$ is performed only once per training step. The structural update $\nabla_{\mathbf{S}}\mathcal{L}$ is efficiently computed via local element-wise operations—effectively reweighting the dense gradient by the current weight matrix $\mathbf{W}$ and the Softmax Jacobian. Furthermore, the structural scores

are updated at a low frequency, thereby significantly reducing the computational cost.

C. Importance-Aware Neuron Pruning

While the Constant Fan-in constraint effectively optimizes intra-neuron connectivity, it neglects inter-neuron redundancy. Under this constraint, the network inevitably retains "dead" neurons that satisfy the fan-in requirement but contribute negligibly to the overall representational capacity. To address this, we introduce a dynamic Neuron Pruning strategy to identify and excise these redundant units, facilitating resource reallocation and further structured compression.

We define the Importance Score Vector $\mathbf{I}$ for the layer as the aggregated saliency of its active connections. Formally, considering the active topology defined by the mask $\boldsymbol{\psi}$, the importance vector is computed as:

$$\mathbf{I} = (\sigma(\mathbf{S}) \odot |\mathbf{W}| \odot \boldsymbol{\psi})\, \mathbf{1}, \tag{9}$$

where $\sigma(\cdot)$ is the Sigmoid function, $|\mathbf{W}|$ denotes the element-wise weight magnitude, and $\mathbf{1}$ represents a column vector of ones which sums the values along the fan-in dimension.

The theoretical basis of this hybrid metric lies in synergizing topological confidence with physical signal strength. This formulation guarantees that retained neurons possess connections that are both topologically favored by the optimizer and physically capable of transmitting significant information.

Given that the magnitude of neuron importance scores varies significantly across layers due to distinct fan-in sizes and weight distributions, employing a global absolute threshold is suboptimal. Instead, we establish a dynamic local baseline determined by layer-wise statistics. Specifically, we compute the mean importance μ of the current layer and define the pruning threshold as $\alpha \cdot \mu$, where α is a scaling factor. In our framework, we set $\alpha = 0.3$, meaning that a neuron is considered redundant and subsequently pruned only if its contribution falls below this layer-adaptive baseline.

Weight redistribution: Upon identifying redundant neurons based on the computed importance score threshold, rather than discarding the associated parameter budget, we employ a weight redistribution mechanism, as illustrated in the weight redistribution component within the neuron pruning and weight update module of Fig. 2. The connections from pruned neurons are reallocated to the remaining active neurons, effectively transferring computational capacity to more salient structures. This dynamic redistribution ensures that the model's overall representational capability is preserved, even as the topology becomes increasingly sparse.

D. Algorithm and Analysis

Algorithm 1 outlines the execution flow of the POST framework as illustrated in Fig. 2, which coordinates the synchronous optimization of model parameters and network topology. The training process operates on two distinct time scales: continuous parameter optimization and periodic structural evolution. In each iteration, standard weight updates are performed first. Subsequently, the topological structure is updated at a specific interval Δt. During the structural update phase, the structural scores $\mathbf{S}$ are first adjusted based on the computed gradients to explore fine-grained connectivity patterns. Following this, the framework evaluates the neuron importance I_i against a dynamic threshold to execute coarse-grained neuron pruning and weight redistribution. Ultimately, this iterative process drives the model to converge towards an optimal network architecture.

Algorithm 1 POST

Require: Training data $\mathcal{D}$; Initialized weights $\mathbf{W}$; Initialized structural scores $\mathbf{S}$; Update frequency Δt; Pruning threshold factor α; Total steps T_{total}.

Initialize binary mask $\boldsymbol{\psi}$ based on $\mathbf{S}$.
$t \leftarrow 0$.
while $t < T_{total}$ **do**
  Sample minibatch $(\mathbf{x}, \mathbf{t})$ from $\mathcal{D}$.
  Compute sparse output $\mathbf{y} = \sigma((\mathbf{W} \odot \boldsymbol{\psi})\mathbf{x})$ and Loss $\mathcal{L}$.
  Update weights $\mathbf{W}$ using optimizer.
  if $t \bmod \Delta t = 0$ **then**
    Update structural scores $\mathbf{S}$ via shared activation gradients in (6).
    Compute neuron importance vector $\mathbf{I}$ via (9) and the pruning threshold.
    Prune redundant neurons based on the threshold and redistribute weights.
    Update binary mask $\boldsymbol{\psi}$ based on updated $\mathbf{S}$.
  end if
  $t \leftarrow t + 1$.
end while

IV. Experiments

A. Experimental Settings

In this section, we compare our method against recent structured Dynamic Sparse Training methods and unstructured DST approaches.

Datasets: We evaluate our proposed framework on three standard benchmark datasets: CIFAR-10 and CIFAR-100 [11], and the large-scale ImageNet-1K dataset [12].

Models: To demonstrate the versatility and scalability of POST, we conduct experiments on diverse network architectures, covering both Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs). For CNNs, we utilize VGG-19 [13], ResNet-32, and ResNet-50 [14]. For Vision Transformers, we employ ViT-S/16 and ViT-B/16 [15].

For the CIFAR-10 and CIFAR-100 datasets, we use a batch size of 128 and set the initial learning rate to 0.1. For the ImageNet-1K dataset, we employ a batch size of 512 and a learning rate of 0.2, accompanied by a linear warm-up phase of 5 epochs.

B. Performance on CIFAR-10/100

As presented in Table I and Table II, POST demonstrates superior performance across diverse architectures on the CIFAR datasets.

TABLE I
COMPARISON OF TOP-1 ACCURACY (%) ON CIFAR-10 AND CIFAR-100 FOR VGG19 AND RESNET-32.

Model	Method	Structured	CIFAR-10 Sparsity			CIFAR-100 Sparsity		
			90%	95%	98%	90%	95%	98%
VGG19	RigL	No	93.77	92.75	90.87	71.34	69.21	65.02
	SRigL	Yes	93.54	92.71	90.83	71.21	69.01	64.79
	POST(Ours)	Yes	**94.01**	**93.23**	**91.31**	**72.05**	**70.11**	**65.88**
	Dense			94.23			74.27	
ResNet-32	RigL	No	93.55	92.39	90.22	70.62	68.47	64.14
	SRigL	Yes	93.49	92.28	90.03	70.53	68.21	63.89
	POST(Ours)	Yes	**94.01**	**92.91**	**91.01**	**71.12**	**69.03**	**64.92**
	Dense			94.89			74.91	

TABLE II
COMPARISON OF TOP-1 ACCURACY (%) ON CIFAR-10 AND CIFAR-100 FOR VIT-S/16.

Model	Method	Structured	CIFAR-10 Sparsity			CIFAR-100 Sparsity		
			80%	90%	95%	80%	90%	95%
ViT-S/16	RigL	No	90.58	88.45	84.56	66.54	65.31	62.32
	DSB	Yes	88.39	83.52	80.14	66.05	63.23	60.08
	SRigL	Yes	89.32	86.88	81.54	66.21	64.81	61.11
	DynaDiag	Yes	89.43	87.09	82.76	66.31	65.06	61.43
	POST(Ours)	Yes	**90.11**	**88.12**	**83.31**	**66.51**	**65.22**	**62.02**
	Dense			89.67			66.61	

First, POST consistently outperforms SRigL on both VGG19 and ResNet-32. Notably, POST not only maintains a lead among structured methods but also surpasses the unstructured baseline RigL in multiple configurations. For instance, on the CIFAR-10 dataset, VGG19 at 90% sparsity achieves an accuracy of 94.01% with POST, outperforming SRigL and RigL by 0.47% and 0.24%, respectively. Similarly, on CIFAR-100, ResNet-32 at 98% sparsity attains an accuracy of 64.92%, significantly exceeding the 63.89% achieved by SRigL.

Second, for the more challenging Vision Transformer architecture, POST achieves the highest Top-1 accuracy across all sparsity levels compared to DSB, SRigL, and the recent DynaDiag. Specifically, at a high sparsity of 95% on CIFAR-100, POST reaches 62.02%, surpassing DynaDiag by 0.59% and SRigL by 0.91%.

TABLE III
COMPARISON OF TOP-1 ACCURACY (%) ON IMAGENET-1K FOR RESNET-50.

Model	Method	Structured	Top-1 Accuracy (%)	Top-1 Accuracy (%)
	Sparsity		80%	90%
ResNet-50	SET	No	72.98	69.71
	DeepR	No	71.70	70.20
	DSR	No	73.24	71.57
	Top-KAST	No	74.76	70.42
	MEST	No	75.39	72.58
	RigL	No	74.98	72.81
	SRigL	Yes	75.01	72.71
	POST (Ours)	Yes	**75.31**	**72.92**
	Dense		76.74	

C. Performance on ImageNet

To validate the effectiveness of the POST framework, we extend our evaluation to the large-scale ImageNet-1K dataset. The results demonstrate that POST is applicable to diverse architectures—ranging from standard CNNs to modern Vision Transformers—and consistently outperforms existing structured dynamic sparse methods, effectively narrowing the performance gap with unstructured approaches.

As shown in Table III, on ResNet-50, POST surpasses the structured baseline SRigL at both 80% and 90% sparsity levels. Notably, at 90% sparsity, POST achieves an accuracy of 72.92%, which not only exceeds SRigL (72.71%) but also outperforms unstructured pruning methods such as RigL (72.81%) and MEST (72.58%). This indicates that POST effectively preserves critical feature extraction pathways within large-scale CNNs.

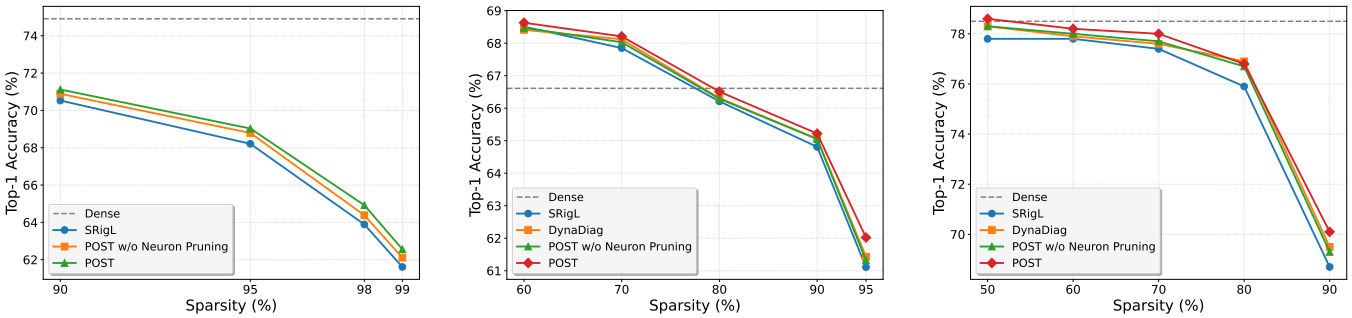
For Vision Transformer models, as presented in Table IV, POST achieves superior performance across the majority of sparsity settings. At 50% sparsity, POST achieves an accuracy of 78.6%, slightly surpassing the dense model (78.5%). Furthermore, at 90% sparsity, it attains 70.1%, outperforming DynaDiag (69.5%) and SRigL (68.7%), thereby validating the versatility of the POST framework.

D. Training and inference speed analysis

Regarding training efficiency, although explicitly optimizing mask scores introduces computational overhead compared to simple magnitude- or gradient-based heuristic rules, our framework effectively mitigates this burden through a synergistic design. By leveraging the Shared Activation Gradient mechanism, we directly reuse the backpropagated error signals

TABLE IV
COMPARISON OF TOP-1 ACCURACY (%) ON IMAGENET-1K FOR VIT-B/16.

Model	Method	Structured	Sparsity				
			50%	60%	70%	80%	90%
ViT-B/16	SET	No	78.2	78.0	77.8	77.0	71.5
	RigL	No	79.8	79.3	78.7	77.2	71.5
	CHTs	No	79.9	79.4	79.0	77.5	71.6
	DSB	Yes	78.0	77.9	76.3	72.9	64.2
	SRigL	Yes	77.8	77.8	77.4	75.9	68.7
	DynaDiag	Yes	78.3	77.9	77.6	**76.9**	69.5
	POST (Ours)	Yes	**78.6**	**78.2**	**78.0**	76.8	**70.1**
	Dense		78.5				



(a) ResNet-32 CIFAR-100 (b) ViT-S/16 CIFAR-100 (c) ViT-B/16 ImageNet

Fig. 3. Ablation study of the proposed neuron pruning method.

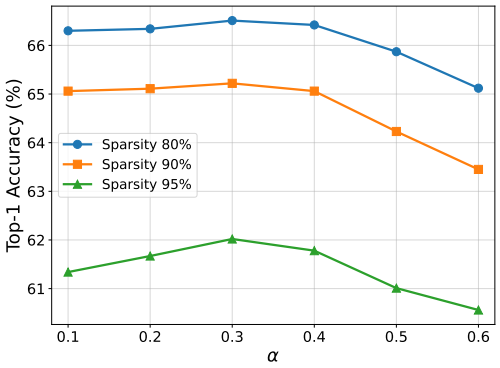


Fig. 4. Ablation study of the hyperparameter α on neuron pruning with ViT-S/16 on CIFAR-100.

from the weight update phase. Furthermore, our method adopts a lower topology update frequency compared to methods like SRigL, reducing computational complexity. Meanwhile, rather than updating all structural scores, we set a search space of the same size as the sparsity, updating only the scores within this search space and the preserved structures. This drastically reduces the computational load, making our computational efficiency comparable to SRigL.

Regarding inference efficiency, the constant fan-in structural sparsity generated by our method aligns with the N:M sparsity pattern, making it fully compatible with modern hardware acceleration, specifically NVIDIA Sparse Tensor Cores. This compatibility significantly reduces memory bandwidth usage and latency, thereby achieving practical acceleration effects typically unattainable by unstructured pruning methods.

E. Impact of Neuron Pruning

To validate the contribution of the neuron pruning component within our framework, we conduct an ablation study by comparing the full POST method against a variant without the neuron pruning mechanism (denoted as "POST w/o Neuron Pruning"). As illustrated in Fig. 3, the quantitative results across ResNet-32, ViT-S/16, and ViT-B/16 demonstrate the significant benefits of this strategy. Observing the performance curves, the full POST framework consistently outperforms the variant without neuron pruning across all sparsity levels. This performance gap becomes particularly pronounced in high sparsity regimes (e.g., $> 95\%$ for CIFAR and 90% for ImageNet). For instance, in Fig. 3(c), at 90% sparsity on ImageNet, the integration of neuron pruning leads to a significant accuracy improvement compared to merely maintaining a sparse topology without removing redundant neurons. Crucially, neuron pruning not only eliminates low-contribution neurons but also reallocates their connection budgets to the remaining active neurons, thereby significantly enhancing the network's overall performance.

F. Impact of Neuron pruning threshold factor α

We further investigate the sensitivity of the hyperparameter α, which controls the aggressiveness of the neuron pruning process. As shown in Fig. 4, the Top-1 accuracy for ViT-S/16 on CIFAR-100 under different sparsity levels (80%, 90%, and

95%) exhibits a consistent inverted U-shaped trend, peaking at $\alpha = 0.3$. A lower threshold results in overly conservative pruning, causing redundant neurons to occupy the topological structure and limiting the benefits of budget reallocation; conversely, an excessively high threshold leads to the erroneous removal of informative neurons, thereby causing a sharp decline in performance. Consequently, we empirically select $\alpha = 0.3$ as the optimal trade-off point to maximize parameter efficiency while preserving representational capacity.

V. Related Work

A. Neural Network Pruning

Neural network pruning methods can be divided into three paradigms based on the training stage. Post-training pruning [16]–[18] typically follows a "pretrain-prune-finetune" pipeline, that is, removing redundant weights from a converged dense model based on magnitude or activation criteria. To eliminate pre-training overhead, Pruning at Initialization (PaI) [19]–[22] attempts to identify the optimal sparse mask before training starts by calculating saliency scores such as gradient flow or synaptic flow. Different from static pruning, pruning during training [23]–[25] optimizes the network topology and weights simultaneously from scratch, allowing the sparse structure to adaptively evolve throughout the entire training process.

B. Dynamic Sparse Training

Dynamic Sparse Training (DST) is a methodology that dynamically adjusts the sparse network structure during the training process. Early works such as SET [23] introduced a "magnitude-pruning, random-growth" cycle, demonstrating that sparse topologies could be learned adaptively during training. DeepR [26] achieved dynamic rewiring by combining stochastic parameter noise with regularization. DSR [27] further refined this approach by introducing dynamic sparse reparameterization with adaptive thresholds, enabling the automatic adjustment of layer-wise sparsity levels. Top-KAST [28] introduced the concept of maintaining dense latent weights, where gradients are applied to the entire latent space while only the top-K active weights participate in the forward pass. MEST [29] proposed pruning by combining the magnitude of existing weights with gradient information. A significant breakthrough was achieved with RigL [3], which replaced random regrowth with a gradient-based criterion. By activating connections with the highest gradient magnitudes, RigL effectively utilizes optimization signals to identify missing critical paths. Building on this, RigL-ITOP [30] analyzed the phenomenon of topology stagnation and proposed utilizing "In-Time Over-Parameterization" to boost the plasticity of sparse networks. BiDST [4] formulated the problem within a bi-level optimization framework to further optimize the sparse network structure. More recently, CHTs [31] employed a network-topology-based approach during the regrowth phase, achieving superior performance.

C. Structured Dynamic Sparse Training

Although the aforementioned methods achieve high theoretical compression rates, they typically produce unstructured sparse patterns. These irregular patterns lead to discontinuous memory access, making it difficult to achieve actual acceleration on general-purpose hardware. DSB [32] introduced a Dynamic Shuffled Block strategy, arranging sparse weights into dense blocks to utilize Tensor Cores for acceleration. SRigL [5] inherits the gradient-based criterion from RigL, utilizing magnitude and gradient scores to update the mask, and subsequently enforces N:M constraints to enable hardware acceleration. Subsequently, DynaDiag [8] proposed imposing diagonal structural constraints, restricting connections within diagonal bands to improve training efficiency and model accuracy.

D. N:M Structured Sparsity

To bridge the gap between theoretical compression rates and practical hardware acceleration, the N:M fine-grained structured sparsity pattern has been introduced, specifically tailored for the NVIDIA Ampere architecture [6], [33]. This pattern enforces a strict constraint requiring that at least N out of every M consecutive weights be zero (e.g., 2:4 sparsity), thereby enabling significant speedups via Sparse Tensor Cores. Furthermore, Schultheis & Babbar [9] demonstrated that implementing a constant fan-in constraint—a special case of N:M sparsity—can yield efficient memory access patterns and enhance throughput on general-purpose GPUs.

VI. Conclusion

In this paper, we proposed POST, a novel framework that reformulates topology search from discrete heuristic selection into continuous probabilistic optimization. This enables the network to utilize structural score gradients rather than proxy metrics to guide the search, ensuring that every structural update strictly follows the direction of loss minimization. Simultaneously, we adopted a hardware-friendly hybrid topology evolution strategy that balances flexibility with hardware efficiency. Our experiments demonstrate that POST outperforms state-of-the-art structured dynamic sparse training methods in accuracy across various modern architectures. In future work, we plan to extend the application of POST to larger-scale foundation models, such as Large Language Models (LLMs), to verify its scalability within massive parameter spaces.

References

[1] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[2] B. Cottier, "Trends in the dollar training cost of machine learning systems," *Epoch. January*, vol. 31, p. 2023, 2023.

[3] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen, "Rigging the lottery: Making all tickets winners," in *International conference on machine learning*. PMLR, 2020, pp. 2943–2952.

[4] J. Ji, G. Li, L. Yin, M. Qin, G. Yuan, L. Guo, S. Liu, and X. Ma, "Advancing dynamic sparse training by exploring optimization opportunities," in *Forty-first International Conference on Machine Learning*, 2024.

[5] M. Lasby, A. Golubeva, U. Evci, M. Nica, and Y. Ioannou, "Dynamic sparse training with structured sparsity," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

[6] A. Mishra, J. A. Latorre, J. Pool, D. Stosic, D. Stosic, G. Venkatesh, C. Yu, and P. Micikevicius, "Accelerating sparse deep neural networks," *arXiv preprint arXiv:2104.08378*, 2021.

[7] X. Zhou, W. Zhang, H. Xu, and T. Zhang, "Effective sparsification of neural networks with global sparsity constraint," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3599–3608.

[8] A. Tyagi, A. Iyer, W. H. Renninger, C. Kanan, and Y. Zhu, "Dynamic sparse training of diagonally sparse networks," *arXiv preprint arXiv:2506.11449*, 2025.

[9] E. Schultheis and R. Babbar, "Towards memory-efficient training for extremely large output spaces–learning with 670k labels on a single commodity gpu," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2023, pp. 689–704.

[10] C. J. Maddison, D. Tarlow, and T. Minka, "A* sampling," *Advances in neural information processing systems*, vol. 27, 2014.

[11] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[16] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Stabilizing the lottery ticket hypothesis," *arXiv preprint arXiv:1903.01611*, 2019.

[17] E. Frantar and D. Alistarh, "Sparsegpt: Massive language models can be accurately pruned in one-shot," in *International conference on machine learning*. PMLR, 2023, pp. 10 323–10 337.

[18] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," *arXiv preprint arXiv:2306.11695*, 2023.

[19] N. Lee, T. Ajanthan, and P. H. Torr, "Snip: Single-shot network pruning based on connection sensitivity," *arXiv preprint arXiv:1810.02340*, 2018.

[20] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," *arXiv preprint arXiv:2002.07376*, 2020.

[21] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," *Advances in neural information processing systems*, vol. 33, pp. 6377–6389, 2020.

[22] Y. Wang, D. Li, and R. Sun, "Ntk-sap: Improving neural network pruning by aligning training dynamics," *arXiv preprint arXiv:2304.02840*, 2023.

[23] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta, "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature communications*, vol. 9, no. 1, p. 2383, 2018.

[24] S. Liu, T. Chen, X. Chen, Z. Atashgahi, L. Yin, H. Kou, L. Shen, M. Pechenizkiy, Z. Wang, and D. C. Mocanu, "Sparse training via boosting pruning plasticity with neuroregeneration," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9908–9922, 2021.

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

[26] G. Bellec, D. Kappel, W. Maass, and R. Legenstein, "Deep rewiring: Training very sparse deep networks," *arXiv preprint arXiv:1711.05136*, 2017.

[27] H. Mostafa and X. Wang, "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4646–4655.

[28] S. Jayakumar, R. Pascanu, J. Rae, S. Osindero, and E. Elsen, "Top-kast: Top-k always sparse training," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 744–20 754, 2020.

[29] G. Yuan, X. Ma, W. Niu, Z. Li, Z. Kong, N. Liu, Y. Gong, Z. Zhan, C. He, Q. Jin *et al.*, "Mest: Accurate and fast memory-economic sparse training framework on the edge," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 838–20 850, 2021.

[30] S. Liu, L. Yin, D. C. Mocanu, and M. Pechenizkiy, "Do we actually need dense over-parameterization? in-time over-parameterization in sparse training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6989–7000.

[31] Y. Zhang, J. Zhao, W. Wu, Z. Liao, U. Michieli, and C. V. Cannistraci, "Brain-inspired sparse training enables transformers and llms to perform as fully connected," in *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*, 2025.

[32] P. Jiang, L. Hu, and S. Song, "Exposing and exploiting fine-grained block structures for fast and accurate sparse training," *Advances in Neural Information Processing Systems*, vol. 35, pp. 38 345–38 357, 2022.

[33] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, "Nvidia a100 tensor core gpu: Performance and innovation," *IEEE Micro*, vol. 41, no. 2, pp. 29–35, 2021.