Structured Directional Pruning via Perturbation Orthogonal Projection

Xiaofeng Liu ©, Qing Wang ©, Member, IEEE, Yunfeng Shao ©, Member, IEEE, Yanhui Geng ©, Senior Member, IEEE, and Yinchuan Li ©, Member, IEEE

Abstract—Despite the great potential of artificial intelligence (AI), which promotes machines to mimic human intelligence in performing tasks, it requires a deep/extensive model with a sufficient number of parameters to enhance the expressive ability. This aspect often hinders the application of AI on resource-constrained devices. Structured pruning is an effective compression technique that reduces the computation of neural networks. However, it typically achieves parameter reduction at the cost of non-negligible accuracy loss, necessitating fine-tuning. This paper introduces a novel technique called Structured Directional Pruning (SDP) and its fast solver, Alternating Structured Directional Pruning (AltSDP). SDP is a general energy-efficient coarse-grained pruning method that enables efficient model pruning without requiring fine-tuning or expert knowledge of the desired sparsity level. Theoretical analysis confirms that the fast solver, AltSDP, achieves SDP asymptotically after sufficient training. Experimental results validate that AltSDP reaches the same minimum valley as the vanilla optimizer, namely stochastic gradient descent (SGD), while maintaining a constant training loss. Additionally, AltSDP achieves state-of-the-art pruned accuracy integrating pruning into the initial training process without the need for fine-tuning. Consequently, the newly proposed SDP, along with its fast solver AltSDP, can significantly facilitate the development of shrinking deep neural networks (DNNs) and enable the deployment of AI on resource-constrained devices.

Index Terms—Pruning, group Lasso, shrinking deep neural networks (DNN), tiny artificial intelligence (tiny-AI).

I. INTRODUCTION

D eep Neural Networks (DNNs) have experienced rapid development in recent years, primarily due to their state-of-the-art performance in various domains [1], [2], [3]. However, the advancement of DNNs is accompanied by certain heuristics, such as the adoption of deeper and more extensive models, which has been a prevailing trend in recent years [4], [5]. These heuristics aim to enhance the expressive ability of neural networks through overparameterization [6]. However, they also pose limitations on their applicability to resource-limited devices, including mobile phones, autonomous cars, and augmented reality devices. Consequently, there is a growing demand for techniques that can effectively shrink DNNs while maintaining their accuracy.

Sparse DNNs are widely used for shrinking deep neural networks (DNNs) due to their reduced memory/capacity requirements and faster inference times [7]. Magnitude pruning is an effective approach to obtain sparse DNNs. Depending on the utilization of neural network structure, magnitude pruning can be categorized into two types: unstructured pruning (fine-grained pruning) and structured pruning (coarse-grained pruning) [8], [9], [10], [11], [12]. Unstructured pruning involves directly pruning weights independently within each layer, achieving higher sparsity while maintaining accuracy. However, it often requires specialized hardware or software accelerators for irregular memory access, which can impact online inference efficiency [11]. On the other hand, structured pruning eliminates structured weights, such as 2D kernels, filters, or entire layers, without necessitating dedicated hardware or software packages. This approach avoids irregular memory access concerns. Unfortunately, structured pruning still faces some challenges. After removing the entire structure of the network, fine-tuning is typically required to achieve better performance [9], which adds extra time and computational intensity along with increased energy consumption [10]. Additionally, these structured pruning methods are often tailored to specific network structures, such as filters or kernels, and lack flexibility in handling heterogeneous structures [11].

Inspired by the benefits of group lasso regularization in various applications like compressed sensing, online learning, and tiny AI [13], [14], [15], this paper introduces a novel structured pruning approach along with an asymptotic solver. Unlike traditional methods, our technique does not require separate fine-tuning, making it highly versatile for handling heterogeneous structures. It enables simultaneous pruning of blocks and other structures, offering greater flexibility. The main contributions of this paper are as follows:
• We propose structured directional pruning (SDP), a general coarse-grained pruning scheme based on group Lasso, enabling to prune model with negligible accuracy loss up. In particular, we orthogonally project the sparse perturbations onto a constant loss value plane and update the network accordingly. Hence, our structured directional pruning suppresses only the unimportant parameters with $s_i > 0$ ($s_i$ defined in Definition 1) and encourages the important ones with $s_i < 0$ simultaneously, while traditional structured pruning methods tend to suppress all parameters, resulting in performance losses and requirements of a separate fine-tuning.

• In addition, a fast implementation solver, named alternating structured directional pruning (AltSDP) algorithm, based on regularized dual averaging is proposed, which can quickly adjust the weights on each structural unit to achieve orthogonal projection. Moreover, theoretical analysis further proves that AltSDP achieves the effect of the directional pruning after sufficient training. Meanwhile, AltSDP can be flexibly combined with many optimizers and algorithms (for example stochastic gradient descent (SGD) with momentum algorithm), which allows AltSDP to be easily applied to various datasets and networks.

• Experimental results on two benchmarks demonstrate that AltSDP without fine-tune achieves high pruning accuracies on both the classic pruning tasks and difficult large FLOPs reduction task.

The remainder of the paper is organized as follows. Section II reviews the basic structured pruning framework and the major related works. In Section III, we explore the potential better structured pruning scheme and proposed structured directional pruning (SDP). The fast solver named alternating structured pruning (AltSDP) are formulated in Section IV achieve the effect of structured directional pruning under some reasonable assumptions. Section V presents the experimental results, followed by some visualized analysis. Finally, conclusion and discussions are given in Section VI.

The main notations used are listed below.

| Notation   | Definition                                                                 |
|------------|---------------------------------------------------------------------------|
| $w^0$      | The initial model parameters                                              |
| $w^*$      | A minimizer of the objective function                                      |
| $\mathcal{G}$ | A structured partition of $\{1, 2, \ldots, d\}$                         |
| $\lambda$  | Hyperparameter for sparse regularization                                  |
| $\mathcal{P}$ | The constant loss value plane around $w^*$                                |
| $s_i$      | Direction factor, $i = 1, \ldots, |\mathcal{G}|$                  |
| $f(\cdot)$ | Loss function, $\mathbb{R}^d \rightarrow \mathbb{R}$                     |
| $S_I(\cdot)$ | A sparse regularization                                                   |
| $\langle \cdot \rangle$ | Projecting the input vector onto the subspace $\mathcal{P}$             |
| $E(\cdot)$ | The normalization operator                                                |

II. BACKGROUND AND RELATED WORK

A. Structured Pruning

Considering a deep neural network with overparameterization $w \in \mathbb{R}^d$, the structured pruning aims to eliminate redundant parameters in $w$ structurally, which can be formulated as

Structured pruning: $\arg \min_{w} f(w) + S(w, \mathcal{G}),$  

where $f(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}$ being the loss function; $\mathcal{G}$ is a structured partition of $\{1, 2, \ldots, d\}$ that used to divide/structure $w$ into $|\mathcal{G}|$ groups or vectors, e.g., $\mathcal{G} = \{\{1, 2, 3\}, \ldots, \{d-1, d\}\}$; $S$ is a sparse regularization, e.g., $\ell_0$ norm, to utilize the structured sparsity of parameters according to $\mathcal{G}$. Taking the group lasso regularization as an example, (1) reduces to

$\arg \min_{w} f(w) + \sum_{i=1}^{|\mathcal{G}|} \lambda \|w_i\|_2,$

where $\lambda > 0$ is a weight factor; $w_i$ is the $i$-th group coefficients of $w$ for $i \in \{1, \ldots, |\mathcal{G}|\}$. We can change $\mathcal{G}$ to achieve different sparse structure, e.g., filter-level sparsity, kernel-level sparsity and vector-level sparsity. And if $|\mathcal{G}| = d$ with $\mathcal{G} = \{\{1\}, \{2\}, \ldots, \{d\}\}$, the structured pruning reduces to non-structured pruning or fine-grained pruning, which prunes weights irregularly.

Note that, the sparse regularization in (2) penalizes all $w_i$ to realize structured pruning, which may increase training loss while pruning. To solve this problem, we propose the structured directional pruning in the next subsection.

B. Related Works

Pruning techniques are effective approaches for model compression, allowing for lighter and more efficient neural networks while maintaining performance. Classical methods such as Optimal Brain Surgeon (OBS) [16] and Optimal Brain Damage (OBD) [17] have significantly influenced the field of structured pruning. In recent years, researchers have made substantial advancements in the pruning domain, introducing various innovative approaches.

Structured pruning: In [18], a network slimming method based on the channel-level sparsity was proposed to automatically identify and prune insignificant channels. In [19], a channel pruning method was proposed via a LASSO regression based channel selection and least square reconstruction. AutoML for model compression was proposed in [20]. Discrimination-aware channel pruning was proposed in [21] to choose channels. In [22], the soft filter pruning was proposed to inference procedure of deep convolutional neural networks. Filter pruning via geometric median was proposed in [23]. Then, Tian et al. proposed a filter pruning method named adding before pruning (ABP) to make the model focus on the filters of higher significance by training [24]. In addition, collaborative channel pruning was proposed in [25]. Lin et al. encoded the second-order information of pretrained weights with a simple fine-tuning procedure [26]. Polarization regularizer was proposed in [27] to suppress only unimportant neurons while keeping important neurons intact. Moreover, correlation-based pruning was proposed in [28]. Unfortunately, the above
methods still suffer from a loss of accuracy when pruning. Fine-tuning is hard to avoid, which requires extra effort and more intensive computing.

**Directional pruning:** The generalized regularized dual averaging (gRDA) algorithm is first adopt to achieve the effect of unstructured directional pruning in [7], in which fine-tuning or the expert knowledge on the sparsity level is no longer needed. However, it requires dedicated hardware or software to accelerate access to irregular memory, affecting the efficiency of online reasoning [11]. Moreover, the learning rate schedule usually applied jointly with the SGD with momentum does not work well for gRDA [7], which means it can’t be flexibly combined with many optimizers, hence hindering its application in advanced AI algorithms. This motivates us to propose structured directional pruning (SDP) and its fast solver alternating structured directional pruning (A-SDP).

### III. SDP: STRUCTURED DIRECTIONAL PRUNING

#### A. SDP: Scheme

Structured directional pruning (SDP) tends to prune the neural network along a direction that does not change the training loss. The idea behind this is first to find a subspace, called $\mathcal{P}$ (red subspace in Fig. 1), where the training loss is fixed, and then project the sparse perturbation onto it. The network is updated with the perturbation after projection to keep the training loss constant.

To find $\mathcal{P}$, we first analyze the local geometry of the loss function through its Hessian matrix. Since $\nabla f(w^\star) \approx 0$, the Hessian $\nabla^2 f(w^\star)$ has multiple nearly zero eigenvalues [29], [30]. According to the second Taylor expansion of $f(w^\star)$, the training loss will be almost constant when pruning in directions related to these eigenvalues. This means that the subspace $\mathcal{P}$ can be generated based on these directions. Note that, traditional structured pruning (the purple vector in Fig. 1) is difficult to prune networks along $\mathcal{P}$, since it is nearly orthogonal to $w^\star$ [29], which may reveal why traditional structured pruning requires fine-tuning.

To prune $w^\star$ along $\mathcal{P}$, inspired by the directional pruning [7], we first introduce direction factors $s_i$, $i = 1, \ldots, |\mathcal{G}|$, which reflects the angle between $w_i$ and $\Pi_i(w^\star)$, where $\Pi_i(\cdot)$ represents an operator of projecting the input vector onto the subspace $\mathcal{P}$, and $\Pi_i(\cdot)$ denotes its $i$-th group that is separated w.r.t. $\mathcal{G}$. Different from (2), structured directional pruning, defined in Definition 1, decrease the magnitude of $w_i^\star$ with $s_i > 0$ (acute angle) and simultaneously increase the magnitude of $w_i^\star$ with $s_i < 0$ (obtuse angle). And we minimize $\frac{1}{2}\|w^\star - w\|^2$ instead of the fixed training loss to limit the optimization range.

**Definition 1 (Structured directional pruning):** Suppose that $w^\star \in \mathbb{R}^d$ is a minimizer satisfies $\nabla f(w^\star) = 0$ with $f(\cdot)$ being the loss function. Assume that none of the coefficients in $w^\star$ is zero. The structured directional pruning is given by

$$
\arg\min_w \frac{1}{2}\|w^\star - w\|^2 + \lambda \sum_{i=1}^{|\mathcal{G}|} s_i \|w_i\|^2,
$$

where $\lambda > 0$ is a weight factor, $\mathcal{G}$ is the structured partition, and $s_i$ is the direction factor

$$
s_i := (E(w_i^\star), \Pi_i(E_G(w^\star))) ,
$$

where $\langle \cdot, \cdot \rangle$ being the inner product, $E(\cdot)$ being the normalization operator, i.e., $E(w) = w/\|w\|_2$, and $E_G(\cdot)$ being the normalization operator w.r.t. $\mathcal{G}$, i.e., $E_G(w) = [E(w_1)^T, E(w_2)^T, \ldots, E(w_{|\mathcal{G}|})^T]^T$.

Our structured directional pruning can also be understood as being based on the orthogonal projection of perturbations. That is, structured pruning in (1) can be rewritten as a perturbation of $w^\star$, i.e.,

$$
w_i^\star - \xi_i E(w_i^\star), \quad i = 1, \ldots, |\mathcal{G}|,
$$

where $0 \leq \xi_i \leq \|w_i^\star\|_2$ and the $i$-th group of $w^\star$ is pruned if $\xi_i = \|w_i^\star\|_2$.

Since usually

$$
[\xi_1 E(w_1^\star)^T, \xi_2 E(w_2^\star)^T, \ldots, \xi_{|\mathcal{G}|} E(w_{|\mathcal{G}|}^\star)^T]^T \notin \mathcal{P},
$$

fine-tuning is need for traditional structured pruning. By comparison, our SDP can be viewed as pruning $w^\star$ by setting $\xi_i = \lambda s_i$, then we have $w_i^\star - \lambda \Pi_i(E_G(w^\star))$, $i = 1, \ldots, |\mathcal{G}|$ by noting that $s_i E(w_i^\star) = \Pi_i(E_G(w^\star))$, i.e., pruning along $\mathcal{P}$ by using the orthogonal projection of perturbations.
B. SDP: Theoretical Analysis

In this subsection, we present the optimal solution of the structured directional pruning in (3). Since the objective function in (3) is separable for each group/structure, we propose the following Theorem 1 to demonstrate that each subproblem has an explicit solution.

**Theorem 1:** Consider the optimization problem

\[
\arg\min_{w_i} \left\{ \frac{1}{2} \| w_i^* - w_i \|^2 + \lambda s_i \| w_i \|_2 \right\}.
\]

For \( w_i^* \in \mathbb{R}^d \setminus \{0\}, s_i \in \mathbb{R}, \lambda > 0 \), (7) has an explicit solution:

\[
\hat{w}_i = \left(1 - \frac{\lambda s_i}{\| w_i^* \|_2} \right) w_i^*.
\]

The above theorem gives the solution to the subproblem of (3), which is proved in Appendix. Since the original problem is a superposition of subproblems, i.e., the above problem indirectly elucidates the solution of directional pruning. However, determining \( s_i \) in Theorem 1 requires computing the Hessian \( \nabla^2 f(w^*) \) to find \( \mathcal{P}_0 \), which is computationally cumbersome. We hence propose a fast solver, named AltSDP, to asymptotically obtain the structural sparse model in the next section. The idea behind is to separate the progress of finding the optimal \( w^* \) and its sparse structure through an alternative manner.

IV. AltSDP: Alternating Structured Directional Pruning

Considering SDP is computationally unfriendly to neural networks, we then propose its fast solver alternating structured pruning (AltSDP), and further prove that the proposed solver can asymptotically achieve the effect of structured directional pruning under some reasonable assumptions.

A. AltSDP: Scheme

Considering an overparameterized DNN with training data \( Z_n = \{X_i, Y_i\}_{i=1}^n \) and parameters \( w \in \mathbb{R}^d \). Assume that \( h(x; w) \) is the network output, denote \( f(w; Z) := \mathcal{L}(h(X; w); Y) \) with \( \mathcal{L}(h; y) \) being a loss function, e.g. the cross-entropy loss or the mean squared error (MSE) loss. And let \( \nabla f(w, Z) \) be the gradient of \( f(w, Z) \) w.r.t. \( w \). To this end, our AltSDP is given by

\[
\begin{align*}
\mathbf{v}_{n+1} &= \mathbf{v}_n - \gamma \nabla f(w_n; Z_{n+1}) \quad \text{AltSDP-(a)} \\
\mathbf{w}_{n+1} &= \arg \min_{\mathbf{w} \in \mathbb{R}^d} \left\{ \frac{1}{2} \| \mathbf{w}_i \|^2_2 - \mathbf{w}^T \mathbf{v}_{n+1} + g(n, \gamma) \sum_{i=1}^{\left| G \right|} \| \mathbf{w}_i \|_2 \right\} \quad \text{AltSDP-(b)}
\end{align*}
\]

where \( n = 0, 1, \ldots, N - 1 \) is the iteration number; \( Z_{n+1} \in \mathbb{Z} \) is the \( n \)-th training data; \( g(n, \gamma) = c\sqrt{\gamma(n + \mu)^\alpha} \) is the tuning function motivated by [31] with \( c, \mu > 0 \) being two hyperparameters that control the strength of penalization. The iteration of AltSDP can be easily started with a random initialization \( w_0 \). Note that, following the proof of Theorem 1, we can have the solution of (AltSDP-(b)), i.e., for each \( i \in [1, \ldots, |\mathcal{G}|] \), we have

\[
w_{n+1,i} = \left(1 - \frac{g(n, \gamma)}{\| \mathbf{v}_{n+1,i} \|_2} \right) \mathbf{v}_{n+1,i}. \tag{9}
\]

**Remark 1:** Following the analysis in [31], [32], it shows that \( g(n, \gamma) \) is the most important part to achieve the structured directional pruning, where \( (n + \mu)^\alpha \) is used to match the growing magnitude of (AltSDP-(a)). Algorithm 1 presents the pseudocode of AltSDP. If \( g(n, \gamma) = n \gamma \) and \( w_0 = 0 \), our AltSDP reduces to the group lasso based regularized dual averaging (RDA) algorithm [33], and no longer has structured directional pruning ability.

B. AltSDP: Theoretical Analysis

In this subsection, we show AltSDP achieves the structured directional pruning asymptotically based on the stochastic gradient descent, i.e., under the condition that \( w^* = w_{SDP} \). Denote \( G(w) := \nabla f(w) := \mathbb{E}_Z [\nabla f(w; Z)] \), 
\( \mathbb{E}_Z [\nabla^2 f(w; Z)] = 1/|Z| \sum_{i=1}^{|Z|} f(w; Z_i) \). Define \( \Sigma(w) := \mathbb{E}_Z [\nabla^2 f(w; Z)] - \nabla f(w; Z) \nabla f(w; Z)^T \). Define the gradient flow \( w(t) \) to be the solution of the ordinary differential equation (ODE)

\[
\frac{d\Phi(t,s)}{dt} = -H(w(t))\Phi(t,s), \quad \Phi(s, s) = I_d.
\]

Define \( w_{\gamma}(t) := w_{\lfloor t/\gamma \rfloor} \), where \( \lfloor x \rfloor \) denotes the greatest integer not greater than \( x \). Then, we make the following reasonable assumptions.

---

**Algorithm 1:** Alternating Structured Directional Pruning (AltSDP) Algorithm

**Input:** Hyperparameters \( c, \mu > 0 \), learning rate \( \gamma \), training data \( Z \), structured partition \( \mathcal{G} \), maximum iterations \( N \)

**Output:** Pruned model \( w_{SDP} \)

Initial parameters \( w_0 \);

Initial auxiliary variable \( v_0 \leftarrow w_0 \);

for \( n = 0 \) to \( N - 1 \) do

\[
Z_n \leftarrow \text{SelectTrainingData}(n; Z);
\]

\[
v_{n+1} \leftarrow v_n - \gamma \nabla f(w_n; Z_n);
\]

for \( i \in \mathcal{G} \) do

\[
w_{n+1,i} \leftarrow \left(1 - \frac{g(n, \gamma)}{\| v_{n+1,i} \|_2} \right) v_{n+1,i};
\]

end

Rebuild the pruned model \( w_{SDP} \leftarrow w_N \);

return \( w_{SDP} \).

---
Assumption 1: $\nabla f(w) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is continuous on $\mathbb{R}^d$.

Assumption 2: $\Sigma(w) : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}$ is continuous, and $E[\sup_{\|w\| \leq K} \|\nabla f(w, Z)\|] < \infty$ a.s. for any $K$.

Assumption 3: The Hessian matrix $H(w) : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}$ is continuous.

Assumption 4: There exists a non-negative definite matrix $\bar{H}$ such that $\int_0^1 \|H(w(s)) - \bar{H}\| ds \leq \infty$ with $\| \cdot \|$ being the spectral norm, and the eigenspace of $\bar{H}$ associated with zero eigenvalues matches the subspace $\mathcal{P}$.

Assumption 5: $c \int_0^1 \Phi(t, s) \frac{dG(s)t^{c_4}}{ds} ds = o(t^c)$, where $E_G(w(t))$ is defined in (4).

Assumption 6: There exists $\bar{T} > 0$, such that $E_G(w(t)) = E_G(w^{SGD}(t)) = E_G(w(T))$ a.s. for $t > \bar{T}$.

Similar to assumptions in [7], the above assumptions hold empirically and strictly under some conditions. We explain in detail in remarks below.

Remark 2 (Condition (A4)): This condition can be verified on some simple networks under the MSE loss. It is known that once $w(t)$ and SGD converge to the same flat valley of minima, the subspace $\mathcal{P}$ matches the eigenspace of $\bar{H}$ associated with the zero eigenvalues [7]. In addition, [36], [37] showed that $w(t) \rightarrow w^*$ for one hidden layer networks under the teacher-student framework and MSE loss, then the limit $\bar{H} = H(w^*)$ and the condition holds.

Remark 3 (Condition (A5)): This assumption can be understood as that $E_G(w(t))$ is assumed to be mainly restricted in the eigenspace of $H(w(t))$ associated with positive eigenvalues as $t \rightarrow \infty$. Since [29], [38] proved that $w(t)$ lies mainly in the subspace of $H(w(t))$ associated with positive eigenvalues, and Fig. 2 shows that the angle between $w(t)$ and $E_G(w(t))$ is very small, we can conclude that this assumption holds empirically.

Remark 4 (Condition (A6)): For the MSE loss, we have $w(t) \rightarrow w^*$ under some conditions according to Remark 2, hence $E_G(w(t)) = E_G(w(T))$ holds. For the cross-entropy loss, [33], [39] proved that $\|w(t)/\|w(t)\|_2$ converges to a unique direction when $\|w(t)\|_2 \rightarrow \infty$, which shows that $E_G(w(t))$ stabilizes after a finite time and hence we have $E_G(w(t)) = E_G(w(T))$ holds. In addition, if $\gamma \rightarrow 0$, the deviation between the gradient flow and the SGD is small, and hence $E_G(w(t)) = E_G(w^{SGD}(t))$ holds.

Next, we propose Theorem 2 to show that $\text{AltSDP}$ achieves the structured directional pruning asymptotically, which is proved in Appendix.

**Theorem 2**: Under Assumptions 1–6, suppose $\mu \in (0, 5, 1)$ and $c > 0$, when $\gamma \rightarrow 0$, $\text{AltSDP}$ achieves structured directional pruning based on $w^{SGD}(t)$ asymptotically, i.e., we have for $t > \bar{T}$
\[
w_{\gamma}(t) \overset{d}{=} \arg \min \frac{\|w^{SGD}(t) - w\|_2^2}{2} + \lambda_{\gamma,i} \sum_{i=1}^{\mid G \mid} s_i \|w_i\|^2,
\]
and
\[
w_{\gamma,i}(t) \overset{d}{=} \left(1 - \frac{\lambda_{\gamma,i}}{\|w_i^{SGD}\|_2} + \tilde{s}_i w_i^{SGD}, \ i = 1, \ldots, \mid G \mid, \right.
\]
where $\lambda_{\gamma,i} = c \sqrt{\gamma}t^c; \overset{d}{\approx}$ represents “asymptotic in distribution” under the empirical probability measure of gradients; and $\tilde{s}_i$ satisfies $\lim_{t \rightarrow \infty}\tilde{s}_i - s_i = 0$ for all $i$.

Theorem 2 shows that $\text{AltSDP}$ achieves directional pruning asymptotically after enough training $(t > \bar{T})$ with learning rate $\gamma \rightarrow 0$. This conclusion is crucial for fitting directional structured pruning into neural network training, which avoids computing the Hessian matrix and makes $\text{AltSDP}$ work as fast as the basic SGD. The left side in (12) denotes the finally solution found by $\text{AltSDP}$, while the right side in (12) denotes the optimal solution of structured directional pruning according to Definition 1 and Theorem 1.

V. EXPERIMENTS

In this section, we carry out extensive experiments to evaluate our $\text{AltSDP}$ algorithm, and present the evidence that $\text{AltSDP}$ achieves the structured directional pruning asymptotically. We compare different structured pruning algorithms in Section V-B. In Section V-C, we analyze the effect of hyper-parameters in $\text{AltSDP}$ and show that it performs the structured directional pruning by checking whether the $\text{AltSDP}$ algorithm reaches the same valley as the SGD algorithm. The source code to replicate the experiments will be available online soon at https://github.com/jluxiaofeng/Structured-Directional-Pruning-via-Perturbation-Orthogonal-Projection.

A. Experimental Setup

We use $\text{AltSDP}$ algorithm to simultaneously train and prune two widely-used deep CNN structures (the VGG-Net [5], and ResNet [2]) on classic datasets (CIFAR-10/100 [43] and ImageNet [44]). Specifically, our method doesn’t need any post-processes like fine-tuning. Our code implementation is based on Pytorch [45].

We compare $\text{AltSDP}$ with different methods that have published results in terms of the test accuracy and Floating-point Operations (FLOPs) reduction. Some methods are reproduced by [27] and obtain better performance than the originally published ones, then we use the better results in our comparisons with appending a label “(New)”. The base ResNet model is implemented following [2], [23], [27] and the base VGG model is implemented following [18], [27]. The detailed parameters for training are list in Appendix. For each method, we present its baseline model accuracy, pruned model accuracy, the accuracy...
drop between baseline and pruned model, and FLOPs reduction after pruning. A negative accuracy drop indicates that the pruned model performances better than its unpruned baseline model. Specifically, the pruned model accuracy reported for our AltSDP is without fine-tuning.

### B. Performance Comparison Results

Table I shows the performance of different methods on CIFAR datasets, which is the most widely used dataset for pruning task. On CIFAR-10, ResNet-56 task, our method obtains the smallest accuracy drop (−0.10%) and the best pruned accuracy (93.90%) with highest FLOPs reduction (55%). On CIFAR-10, VGG-16 task, our method also obtains the smallest accuracy drop (−0.09%) and the best pruned accuracy (93.97%) with highest FLOPs reduction (55%). Since few structured pruning results on CIFAR-100 dataset are reported in previous works, we only compared with three different algorithms. And as shown in Table I, our method still achieves the smallest accuracy drop (−0.06 for ResNet-56 and −0.46 for VGG-16) and the best pruned accuracy (72.55 for ResNet-56 and 74.29 for VGG-16) under similar FLOPs reduction.

Table II shows the performance of different methods on large FLOPs reduction. Since there exist few related works on structured pruning with large FLOPs reduction, the comparison results are mainly from [7]. Our method without fine-tuning achieves the smallest accuracy drop (0.86%) and the best pruned accuracy (92.94%) with the highest FLOPs reduction (72%).

Table III shows that the proposed method outperforms previous methods on large datasets, i.e., ILSVRC2012, without fine-tuning. For ResNet-18, our method achieves the same inference speed up with [22] and [23], but its accuracy exceeds by 0.74% and 0.03% respectively.

### C. Analysis

In this Section, we first empirically study the effect of hyper-parameters in AltSDP. Then, we check whether AltSDP performs structured directional pruning and reaches the same flat minimum valley obtained by SGD. Similar analysis strategy has been done by [7], [47], [48], [49], [50] and the base VGG model and method for visualizing are implemented following [7], [47].

We then displays the performance of SGD and AltSDP with different hyper-parameter c and µ on VGG-16, CIFAR-10 task. As shown in Fig. 3, the training loss of AltSDP is almost the same with SGD (diff. less than 0.003 in Table IV) when pruning, which implies that AltSDP reaches the same flat minimum valley found by SGD. And the test accuracy of AltSDP is similar with SGD. Table IV shows more details of Fig. 3, where sparsity denotes the non-zero parameter ratio after training. We find that AltSDP performs worse than SGD when µ = 0.40, but performs better than SGD under other parameter

---

**TABLE I**

| Method       | Baseline | Pruned |
|--------------|----------|--------|
|             | Acc. (%) | Acc. (%) | Acc. Drop (%) | FLOPs Reduction |
| ResNet-56, CIFAR-10 task | 93.80    | 93.27   | 0.53          | 48% |
| NS [18] (New) | 92.80    | 91.80   | 1.00          | 50% |
| CP [19]      | 92.80    | 91.90   | 0.90          | 50% |
| AMIC [20]    | 93.80    | 93.49   | 0.31          | 50% |
| DCP-adapt [21] | 93.80   | 93.81   | −0.01         | 47% |
| SFP [22]     | 93.59    | 93.35   | 0.24          | 51% |
| FPGM [23]    | 93.59    | 93.49   | 0.10          | 53% |
| CCP [25]     | 93.50    | 93.46   | 0.04          | 47% |
| DeepHoyer [25] | 93.80    | 93.54   | 0.26          | 48% |
| PR [27]      | 93.80    | 93.83   | −0.03         | 47% |
| FilterSketch [26] | 93.26   | 93.19   | 0.07          | 42% |
| ABP [24]     | 93.41    | 93.10   | 0.31          | 45% |
| CORESE [40]  | 93.39    | 92.74   | 0.65          | 50% |
| MFP [41]     | 93.59    | 93.56   | 0.02          | 53% |
| SOKS [42]    | 93.06    | 93.08   | −0.02         | 55% |
| Ours         | 93.80    | 93.90   | −0.10         | 55% |

**TABLE II**

| Method       | Baseline | Pruned |
|--------------|----------|--------|
|             | Acc. (%) | Acc. (%) | Acc. Drop (%) | FLOPs Reduction |
| ResNet-56, CIFAR-100 task | 93.80    | 93.62   | 0.26          | 51% |
| NS [18] (New) | 93.88    | 93.54   | 0.04          | 34% |
| FPGM [23]    | 93.88    | 93.92   | −0.04         | 54% |
| PR [27]      | 93.96    | 93.75   | 0.21          | 54% |
| ABP [24]     | 93.88    | 93.97   | −0.09         | 55% |
| Ours         | 93.88    | 93.97   | −0.09         | 55% |

**TABLE III**

| Method       | Baseline | Pruned |
|--------------|----------|--------|
|             | Acc. (%) | Acc. (%) | Acc. Drop (%) | FLOPs Reduction |
| VGG-16, CIFAR-10 task | 72.49    | 71.40   | 1.09          | 24% |
| NS [18] (New) | 72.49    | 72.46   | 0.06          | 25% |
| PR [27]      | 72.49    | 72.55   | −0.06         | 24% |
| Ours         | 72.49    | 72.55   | −0.06         | 24% |

**TABLE IV**

| Method       | Baseline | Pruned |
|--------------|----------|--------|
|             | Acc. (%) | Acc. (%) | Acc. Drop (%) | FLOPs Reduction |
| MIL [46]     | 69.98    | 66.33   | 3.65          | 35% |
| SFP [22]     | 70.28    | 67.10   | 3.18          | 42% |
| FPGM [23]    | 70.28    | 67.81   | 2.47          | 42% |
| Ours         | 70.28    | 67.84   | 2.44          | 42% |
Fig. 3. Performance comparison results of AltSDP (SDP) and SGD under different hyperparameters on VGG-16, CIFAR-10 task. Left: training loss. Center: training accuracy. Right: testing accuracy.

TABLE IV
RESULTS ON VGG-16, CIFAR-10 TASK: THE EFFECT OF HYPERPARAMETERS

| SGD | Structured directional pruning |
|-----|--------------------------------|
| no other parameters | c = 5e-7 | c = 5e-7 | c = 5e-7 | c = 5e-7 | c = 8e-7 | c = 8e-7 | c = 8e-7 |
| u = 0.40 | u = 0.51 | u = 0.55 | u = 0.60 | u = 0.40 | u = 0.51 | u = 0.55 |
| Train loss | 0.0001 | 0.0002 | 0.0012 | 0.0026 | 0.0022 | 0.0006 | 0.0023 | 0.0027 |
| Test Acc. | 0.9089 | 0.9080 | 0.9091 | 0.9090 | 0.9153 | 0.9077 | 0.9110 | 0.9127 |
| Sparsity | 0.0000 | 0.0000 | 0.0090 | 0.1242 | 0.0084 | 0.4391 | 0.6767 |

Fig. 4. The upper figures show the contour of training loss and testing error on the hyperplane. The lower figures show the corresponding white curve (Bézier curve), which contains the interpolating minimizers of SGD and AltSDP under different $\mu$.

settings. This is reasonable according to Theorem 2, which suggests that $\mu$ should be slightly greater than 0.5. Moreover, as hyperparameters $c$ and $\mu$ become larger, AltSDP pushes more parameters to zero and the sparsity becomes larger.

We check whether AltSDP reaches the same valley found by SGD. We train VGG16 on CIFAR-10 until nearly zero training loss using both SGD and AltSDP. We use the method of [51] to search for a quadratic Bézier curve of minimal training loss connecting the minima found by optimizers. Form Fig. 4, we can see that AltSDP performs the structured directional pruning since the learned parameters of both SGD and AltSDP lie in the same flat minimum valley on the training loss landscape if $\mu$ and $c$ is properly tuned, namely $\mu = 0.55$ and $c = 8 \times 10^{-7}$. More visualized results are showed in Figs. 5–10.

Finally, the experimental results presented in Table V demonstrate the excellent pruning performance of our algorithm with various optimizers, namely AdamW and AMSGrad, where “FR.” represents “FLOPs Reduction.” Comparing the pruned
model to the non-pruned method, we observe only a minor performance decrease of 2.44% for SGD, 1.56% for AdamW, and 1.42% for AMSGrad. These results emphasize the versatility and effectiveness of our proposed technique in conjunction with different optimizers, underscoring its potential for application in a broader range of scenarios beyond SGD. Moreover, Table V presents a comprehensive overview of the overall pruning cost, measured in terms of GPU time consumption by AltSDP and the no-pruning baseline. The training times were tested based on a single GeForce RTX 3090. Notably, the results indicate that, across different optimizers, the time cost of our proposed pruning method remains consistent with the no-pruning baseline. This consistency underscores the superiority of the fast iterative solver AltSDP.

In comparison to alternative pruning methods, many of which adopt a training scheme consisting of 200 epochs of sparse training followed by 200 epochs of fine-tuning on CIFAR/ResNet & VGG tasks [27], our method distinguishes itself by requiring only 200+ epochs of sparse training without subsequent fine-tuning. This streamlined approach effectively reduces the overall training time. It is important to highlight that our average pruning training time per epoch aligns with the no-pruning baseline time, allowing for a fair comparison based on the number of epochs. This reinforces the efficiency gains achieved by our proposed method, demonstrating its effectiveness in significantly reducing training time while maintaining consistent performance.

D. Experimental Setup Details

1) Training Setup in Section V-B: The base ResNet model is implemented following [2], [23], [27], and the base VGG model is implemented following [18], [27]. For our experiments in Tables I–III, we mainly follow the codes of [27] and [23]. The detail hyperparameters to obtain the best results are summarized in Table VI, where the learning rate decay scheme “[60, 160,…]@[0.2, 0.2,…]” means that the learning rate multiplied by 0.2 at 60 epoch and multiplied by 0.2 at 160 epoch, etc. We set nonzero ratio lower bounds 0.3, 0.25, and 0.3 respectively for experiments in Tables I–III to avoid excessive pruning.

2) Training Setup in Section V-C: The base VGG model and method for visualizing are implemented following [7], [47], which does not have batch normalization. For both SGD and AltSDP we use the similar learning rate schedule adopted by [7]: fix the learning rate equals to 0.1 at first 50% epochs, then reduce the learning rate to 0.1% of the base learning rate between 50% and 90% epochs, and keep reducing it to 0.1% for the last 10% epochs. The minibatch size is 128 for all experiments in Section V-C.

3) The Selection of $G$: In our experiments, we utilize neural-level granularity for the fully connected layer and filter-level granularity for the convolution layer during evaluation. The flexibility of choosing $G$ is advantageous, enabling the proposed method’s deployment in heterogeneous structures, encompassing a combination of neuron-level, filter-level, and channel-level pruning. Additionally, the results presented in Table V showcase AltSDP’s versatility in
combining seamlessly with various optimizers. The compatibility with different optimizers, coupled with the diverse choices of $G$, collectively contribute to AltsDP’s enhanced flexibility.

VI. CONCLUSION

In this paper we propose the structured directional pruning method to compress deep neural networks while preserving accuracy, which is based on orthogonal projecting the sparse perturbations onto the flat minimum valley found by optimizers. A fast solver AltsDP is also proposed to achieve structured directional pruning. Theoretically, we prove that AltsDP achieves directional pruning after sufficient training. Experimentally, we demonstrate the benefits of structured directional pruning and show that it achieves the state-of-the-art result. Experiments on two benchmarks show that our method obtains the best pruned accuracy without fine-tuning. Moreover, more visualized results demonstrate our method performs directional pruning, reaching the same minimal valley as the optimizer.

APPENDIX

A. Proof of Theorem 1

Theorem 1. Consider the optimization problem

$$\arg\min_{w_i} \left\{ \frac{1}{2} \| w_i^* - w_i \|_2^2 + \lambda s_i \| w_i \|_2 \right\}. \quad (14)$$

For $w_i^* \in \mathbb{R}^d \setminus \{0\}$, $s_i \in \mathbb{R}$, $\lambda > 0$, (14) has an explicit solution:

$$\hat{w}_i = \left(1 - \frac{\lambda s_i}{\| w_i \|_2} \right) w_i^*. \quad (15)$$

Proof: Let’s $\hat{w}_i$ denotes the solution for (14), we prove $\hat{w}_i$ follows the formulation in (15) from three perspectives: $s_i = 0$, $s_i > 0$ and $s_i < 0$. Set $f(w_i) := \frac{1}{2} \| w_i^* - w_i \|_2^2 + \lambda s_i \| w_i \|_2$.

First, when $s_i = 0$, the solution $\hat{w}_i = w_i^*$. Then, on the one hand, when $s_i > 0$, the objective function is convex, therefore $\nabla f(\hat{w}_i) = 0$. We have

$$\nabla f(\hat{w}_i) = \hat{w}_i - w_i^* + \frac{\lambda s_i \hat{w}_i}{\| w_i \|_2} = 0,$$

which yields $w_i^* = (1 + \lambda s_i/\| w_i \|_2) \hat{w}_i$. Since $1 + \lambda s_i/\| w_i \|_2 > 0$ is a scalar, we have $w_i^*$ and $\hat{w}_i$ in the same direction, hence

$$\frac{\hat{w}_i}{\| w_i \|_2} = \frac{w_i^*}{\| w_i^* \|_2}. \quad (17)$$

By substituting (17) into (16), we finish the proof for $s_i > 0$.

On the other hand, when $s_i < 0$, the objective function is not convex, therefore we need to check the value of $f(w_i)$ at stationary points. If $w_i^* = 0$, then $\hat{w}_i = 0$ is the solution for (14). If $w_i^* \neq 0$, we have

- On $\| w_i \|_2 = -\lambda s_i$, $\nabla f(w_i) = -w_i^* \neq 0$, there is no stationary point.
- On $\| w_i \|_2 > -\lambda s_i$, we have $(1 + \lambda s_i/\| w_i \|_2) > 0$. The stationary point $w_{sp1}$ is $(1 - \lambda s_i/\| w_i \|_2) w_i^*$ with objective function value $f(w_{sp1})$

$$f(w_{sp1}) = \frac{1}{2} \| \lambda s_i w_i^* \|_2^2 + \lambda s_i \left(1 - \frac{\lambda s_i}{\| w_i \|_2} \right) \| w_i^* \|_2.$$

TABLE V

| Optimizer | Method       | Acc.(%)/ 20 Epoch | Acc.(%)/ 50 Epoch | Acc.(%)/ 80 Epoch | Acc.(%)/ 100 Epoch | Time / Epoch |
|-----------|--------------|-------------------|------------------|------------------|-------------------|-------------|
| SGD       | No-Prune     | 49.62             | 64.75            | 69.72            | 70.28             | 16 min      |
|           | Prune (FR. 42%) | 46.68 ▼2.94      | 60.46 ▼4.29      | 66.95 ▼2.77      | 67.84 ▼2.44      | 16 min      |
| AdamW     | No-Prune     | 61.84             | 64.42            | 65.52            | 65.83             | 16 min      |
|           | Prune (FR. 42%) | 59.28 ▼2.56      | 62.63 ▼1.79      | 63.94 ▼1.58      | 64.27 ▼1.56      | 16 min      |
| AMSGrad   | No-Prune     | 62.15             | 64.66            | 65.62            | 65.72             | 16 min      |
|           | Prune (FR. 42%) | 58.90 ▼3.25      | 62.53 ▼2.13      | 63.83 ▼1.79      | 64.30 ▼1.42      | 16 min      |

TABLE VI

| Dataset/Model | Learning Rate | Decay Scheme | Batch Size | Hyper-parameters | Result |
|---------------|---------------|--------------|------------|------------------|--------|
| CIFAR-10/ResNet-56 | 0.05 | [60, 160, 200, 220, 240]@[0.2, 0.2, 0.2, 0.2, 0.4] | 64/260 | $c = 10^{-5}$, $\mu = 0.55$ | Table I |
| CIFAR-10/VGG16 | 0.05 | [60, 160, 200, 220, 240]@[0.2, 0.2, 0.2, 0.2, 0.4] | 64/260 | $c = 10^{-5}$, $\mu = 0.55$ | Table I |
| CIFAR-10/ResNet-56 | 0.05 | [60, 160, 200, 220, 240]@[0.2, 0.2, 0.2, 0.2, 0.4] | 64/260 | $c = 10^{-5}$, $\mu = 0.60$ | Table I |
| CIFAR-10/VGG16 | 0.05 | [60, 160, 200, 220, 240]@[0.2, 0.2, 0.2, 0.2, 0.4] | 64/260 | $c = 10^{-5}$, $\mu = 0.60$ | Table I |
| ILSVRC-2012/ResNet-18 | 0.1 | [30, 60, 90]@[0.1, 0.1, 0.1] | 256/100 | $c = 10^{-7}$, $\mu = 0.60$ | Table III |
| ILSVRC-2012/ResNet-18 | 0.001 | (0.9, 0.999), eps = 1e-08, weight_decay = 0.0001 | 256/100 | $c = 10^{-7}$, $\mu = 0.60$ | Table V (SGD) |
• On \( \| w_i \|_2 < -\lambda s_i \), we have \((1 + \lambda s_i / \| w_i \|_2) w_i^* \) with objective function value \( f(w_{sp2}) \)

\[
f(w_{sp2}) = \frac{1}{2} \left( \frac{\lambda s_i w_i^*}{\| w_i^* \|_2} \right)^2 + \lambda s_i \left( 1 + \frac{\lambda s_i}{\| w_i^* \|_2} \right) \| w_i \|_2.
\]

Since \( \lambda s_i < 0 \) and \( w_i^* \neq 0 \), we have

\[
\left( 1 - \frac{\lambda s_i}{\| w_i^* \|_2} \right) w_i^* > \left( 1 + \frac{\lambda s_i}{\| w_i^* \|_2} \right) w_i^*.
\]

Then \( f(w_{sp1}) < f(w_{sp2}) \), which means the global minimizer of \( f(w_i) \) is the stationary point \((1 - \lambda s_i) w_i^* / \| w_i^* \|_2 \) on \( \| w_i \|_2 > -\lambda s_i \). We finish the proof for \( s_i > 0 \).

Then we complete the proof of Theorem 1. \( \square \)

### B. Proof of Theorem 2

**Theorem 2.** Under Assumptions 1–6, suppose \( \mu \in (0.5, 1) \) and \( c > 0 \), when \( \tau \to 0 \), Alt\text{S}\text{DP} achieves structured directional pruning based on \( w_{SGD}(t) \) asymptotically, i.e., we have for \( t > T \)

\[
w_\gamma(t) \approx \arg \min_{w \in \mathbb{R}^G} \frac{\| w_{SGD}(t) - w \|_2^2}{2} + \lambda_{s,t} \sum_{i=1}^{\| G \|} s_i \| w_i \|_2,
\]

and

\[
w_{\gamma,i}(t) \approx \left( 1 - \frac{\lambda_{s,t,s_i}}{\| w_{SGD}(t) \|^2} \right) w_{SGD}(t), \quad i = 1, \ldots, \| G \|,
\]

where \( \lambda_{s,t} = c \sqrt{t^n} \); \( \approx \) represents “asymptotic in distribution” under the empirical probability measure of gradients; and \( s_i \) satisfies \( \lim_{t \to \infty} |s_i - s_i| = 0 \) for all \( i \).

To proof Theorem 2, we first present the following useful Theorem 3, which is proved in Appendix C.

**Theorem 3:** Suppose (A1), (A2) and (A3) hold, and assume that the root of the coordinates \( w(t) \) occur at time \( \{ T_k \}_{k=1}^\infty \subset [0, \infty) \). Let \( w_0 \) with \( w_0,i \neq 0 \) (e.g. from a normal distribution) and \( T_0 = 0 \). Then, as \( \gamma \) is small, for \( t \in (T_K, T_{K+1}) \),

\[
v_{\gamma}(t) \approx w(t) + \sqrt{\gamma} c t^m E_G(w(t))
\]

\[
- \sqrt{\gamma} c \sum_{k=1}^K \left\{ \Phi(t, T_k) \left[ E_G\left( w(T_k^+ \right) - E_G\left( w(T_k^-) \right) \right] \right\} T_k^1,
\]

\[
- \sqrt{\gamma} c \int_0^t \Phi(t, s) \frac{\partial E_G(w(s))}{\partial s} ds
\]

\[
+ \sqrt{\gamma} \int_0^t \Phi(t, s) \Sigma_1/2(w(s)) dB(s)
\]

where \( \approx \) denotes approximately in distribution, \( B(s) \) is a \( d \)-dimensional standard Brownian motion, \( \Phi(t, s) \in \mathbb{R}^{d \times d} \) is the principal matrix solution of the matrix ODE system,

\[
dx(t) = -H(w(t))x(t)dt, \quad x(t_0) = x_0.
\]

and \( \Sigma(w) \) is defined as

\[
\Sigma(w) := \mathbb{E}_G \left[ \left( \nabla f(w; Z) - G(w)\nabla f(w; Z) - G(w)^T \right) \right].
\]

**Theorem 3** presents the distribution dynamics of \( v_{\gamma,i}(t) \) in \( \text{Alt\text{S}\text{DP}}-(a) \) with \( v_{\gamma}(t) := \mathbb{v}(t/\gamma) \). Next, we start prove Theorem 2, which equals to prove the distribution dynamics of \( w_{\gamma,i}(t) \) in \( \text{Alt\text{S}\text{DP}}-(b) \) approximately in distribution with \( \left( 1 - \frac{\lambda_{s,t,s_i}}{\| w_{SGD}(t) \|_2} \right) w_{SGD} \)

**Proof:** To start with, recall that

\[
v_{n+1} = v_n - \gamma \nabla f(w_n; Z_{n+1}) \quad (\text{Alt\text{S}\text{DP}}-(a))
\]

\[
v_{n+1} = \arg \min_{w \in \mathbb{R}^G} \left\{ \frac{1}{2} \| w \|_2^2 - w^T v_{n+1} + g(n, \gamma) \sum_{i=1}^{\| G \|} \| w_i \|_2 \right\} \quad (\text{Alt\text{S}\text{DP}}-(b))
\]

To analysis the distribution dynamics of \( w_{\gamma,i}(t) \), we first define

\[
\zeta_{\gamma,i}(w_{\gamma,i}(t)) := \frac{1}{2} \| w_{\gamma,i}(t) \|_2^2 + \sqrt{\gamma} c t^m \| w_{\gamma,i}(t) \|_2.
\]

Then we have its Fenchel conjugate is given by \( [31] \)

\[
\zeta_{\gamma,i}(w_{\gamma,i}(t)) := \max_{w_{\gamma,i}(t)} \left\{ w_{\gamma,i}(t)^T v_{\gamma,i}(t) - \frac{1}{2} \| w_{\gamma,i}(t) \|_2^2 \right\},
\]

and the derivative of its Fenchel conjugate is given by \( [31] \)

\[
\nabla \zeta_{\gamma,i}(w_{\gamma,i}(t)) := \arg \min_{w_{\gamma,i}(t)} \left\{ \frac{1}{2} \| w_{\gamma,i}(t) \|_2^2 + \sqrt{\gamma} c t^m \| w_{\gamma,i}(t) \|_2 \right\},
\]

Hence, by noting that \( \gamma \) is a convex function, let the gradient equals to zero we have

\[
w_{\gamma,i}(t) + \sqrt{\gamma} c t^m \| w_{\gamma,i}(t) \|_2 - v_{\gamma,i}(t) = 0,
\]

which yields \( v_{\gamma,i}(t) = (1 + \sqrt{\gamma} c t^m) \| w_{\gamma,i}(t) \|_2 \) \( w_{\gamma,i}(t) \). Since \( (1 + \sqrt{\gamma} c t^m) \| w_{\gamma,i}(t) \|_2 > 0 \) is a scalar, we have \( v_{\gamma,i}(t) \) and \( w_{\gamma,i}(t) \) in the same direction, hence

\[
v_{\gamma,i}(t) = v_{\gamma,i}(t) \| w_{\gamma,i}(t) \|_2 = w_{\gamma,i}(t) \| w_{\gamma,i}(t) \|_2.
\]

By substituting \( (25) \) into \( (24) \), we have

\[
w_{\gamma,i}(t) = \nabla \zeta_{\gamma,i}(v_{\gamma,i}(t)) = v_{\gamma,i}(t) - \sqrt{\gamma} c t^m E_G(v_{\gamma,i}(t)).
\]

Now, we obtain the relationship between \( w_{\gamma,i}(t) \) and \( v_{\gamma,i}(t) \) in \( (26) \). Next, we prove \( (26) \) approximately in distribution with \( \left( 1 - \frac{\lambda_{s,t,s_i}}{\| w_{SGD}(t) \|_2} \right) w_{SGD} \) based on \( \text{Theorem 3} \). Follows by \( (20) \) in \( \text{Theorem 3} \) we have

\[
v_{\gamma}(t) \approx w(t) + \sqrt{\gamma} c t^m E_G(w(t)) - \sqrt{\gamma} \delta(t) + \sqrt{\gamma} U(t),
\]

where \( \delta(t) = \delta_1(t) + \delta_2(t) \) and \( U(t) \) are given by

\[
\delta_1(t) := c \sum_{k=1}^K \left\{ \Phi(t, T_k) \left[ E_G(w(T_k^+)) - E_G(w(T_k^-)) \right] \right\} T_k^\mu
\]

\[
\delta_2(t) := \sqrt{\gamma} \int_0^t \Phi(t, s) \Sigma_1/2(w(s)) dB(s)
\]
\[
\delta_2(t) := c \int_0^t \Phi(t, s) \frac{\partial E^\mu_\Gamma(w(s)) s^\mu}{\partial s} ds, \\
U(t) := \int_0^t \Phi(t, s) \Sigma^{1/2}(w(s)) dB(s).
\]

By substituting (27) into (26), for each \( i \) we have

\[
w_{\gamma, i}(t) \overset{d}{=} w_i(t) + \sqrt{\gamma} U_i(t) + \sqrt{\gamma} \delta_i(t) + \sqrt{\gamma} \delta_i(t),
\]

where the equality follows by (25). Following the analysis in [6], [52], the piecewise constant process of SGD follows

\[
w_j^{SGD}(t) \overset{d}{=} w_j(t) + \sqrt{\gamma} U_j(t), \quad j = 1, 2, \ldots, d,
\]

where \( \gamma \) is the least positive eigenvalue of \( \Phi \).

Following the proof in [7] and Levinson theorem in [53], we obtain

\[
\lambda = \lim_{t \to \infty} \langle w(t) \rangle.
\]

Then we have

\[
w_{\gamma, i}(t) \overset{d}{=} w_i^{SGD}(t) - \sqrt{\gamma} \delta_i(t), \quad i = 1, 2, \ldots, |\mathcal{G}|.
\]

Next we prove for \( t \to \infty \),

\[
\delta(t) = c t^\mu \Pi E^\mu_\Gamma(w(t)) + o(t^\mu) + O(t^{\mu-1}).
\]

To obtain this, we need to find the principal matrix solution \( \Phi(t, s) \) in (21). Recall (11) that

\[
\frac{d \Phi(t, s)}{dt} = -H(w(t)) \Phi(t, s), \quad \Phi(s, s) = I_d.
\]

Following the Levinson theorem in [53], when \( a_t \to 0 \), there exists a real symmetric matrix \( \bar{H} = P \Lambda P^T \) satisfying

\[
\sum_{t=0}^\infty \| H(w(s)) - \bar{H} \| ds = O(a_t),
\]

where \( \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d) \) is a diagonal matrix with non-negative values and \( P \) is an orthonormal matrix with its column vectors are eigenvectors \( u_j \).

Following the proof in [7] and Levinson theorem in [53], we get the principal matrix solution \( \Phi(t, s) \) in (21) satisfies

\[
\Phi(t, s) = P(I_d + O(a_t)) e^{-\Lambda(t-s)} P^T
\]

\[
= P_0 P_0^T + O(e^{-\lambda(t-s)} + O(a_t)),
\]

where \( \lambda \) is the least positive eigenvalue of \( \bar{H} \), the column vectors of \( P_0 P_0^T \) are eigenvectors associated with the zero eigenvalue, i.e., \( P_0 P_0^T = \sum_{j} Hw_j = u_j u_j^T \). Then we have

\[
\delta_1(t) = c \int_0^t \Phi(t, s) \left\{ E^\mu_\Gamma(w(T_k^+)) - E^\mu_\Gamma(w(T_k^-)) \right\} T_k^\mu ds
\]

\[
= c P_0 P_0^T \int_0^t \left\{ E^\mu_\Gamma(w(T_k^+)) - E^\mu_\Gamma(w(T_k^-)) \right\} T_k^\mu ds
\]

\[
+ O(e^{-\lambda(t-T_k)} T_k^\mu) + O(a_t T_k^\mu),
\]

where the first term of \( \delta_1(t) \) can be rewritten as

\[
c P_0 P_0^T \sum_{k=1}^K \left\{ E^\mu_\Gamma(w(T_k^+)) - E^\mu_\Gamma(w(T_k^-)) \right\} T_k^\mu
\]

\[
= c P_0 P_0^T \left\{ E^\mu_\Gamma(w(T_k^+)) T_k^\mu - \int_0^{T_k} \frac{\partial E^\mu_\Gamma(w(s)) s^\mu}{\partial s} ds \right\},
\]

Combining (32) and (31), we get

\[
\delta_1(t) = c P_0 P_0^T \left\{ E^\mu_\Gamma(w(T_k^+)) T_k^\mu - \int_0^{T_k} \frac{\partial E^\mu_\Gamma(w(s)) s^\mu}{\partial s} ds \right\}
\]

\[
+ O(e^{-\lambda(t-T_k)} T_k^\mu) + O(a_t T_k^\mu).
\]

Then by substituting (30) into \( \delta_2(t) \), we obtain

\[
\delta_2(t) = c P_0 P_0^T \int_0^t \frac{\partial E^\mu_\Gamma(w(s)) s^\mu}{\partial s} ds
\]

\[
+ O\left( \int_0^t e^{-\lambda(t-s)} \frac{\partial E^\mu_\Gamma(w(s)) s^\mu}{\partial s} ds \right)
\]

\[
+ O\left( \int_0^t a_t \frac{\partial E^\mu_\Gamma(w(s)) s^\mu}{\partial s} ds \right),
\]

where \( \| E^\mu_\Gamma(w(s)) \| = |\mathcal{G}|^{1/2}. \) Then we have

\[
O\left( \int_0^t e^{-\lambda(t-s)} \frac{\partial E^\mu_\Gamma(w(s)) s^\mu}{\partial s} ds \right)
\]

\[
\leq c \int_0^t e^{-\lambda(t-s)} \left\| \frac{\partial E^\mu_\Gamma(w(s))}{\partial s} \right\| s^\mu ds
\]

\[
= c |\mathcal{G}|^{1/2} \int_0^t e^{-\lambda(t-s)} s^\mu ds
\]

\[
\leq c t^{\mu-1} \int_0^t s^{\mu-1} e^{-\lambda(t-s)} ds \overset{(a)}{=} O(t^{\mu-1}),
\]

where \( (a) \) follows by using the similar arguments as the proof of Theorem 4.2 in [31] with \( \mu \in (0, 1) \). Combining (33), (34) and (35) we get

\[
\delta_2(t) = c P_0 P_0^T \int_0^t \frac{\partial E^\mu_\Gamma(w(s)) s^\mu}{\partial s} ds + O(t^{\mu-1}) + O(a_t t^\mu).
\]

Note that \( a_t t^\mu > a_t T_k^\mu \) for \( \mu > 0 \), \( t > T_k \) and \( e^{-\lambda(t-T_k)} \to 0 \). Then

\[
\delta(t) = \delta_1(t) + \delta_2(t)
\]

\[
= c P_0 P_0^T \int_0^t \frac{\partial E^\mu_\Gamma(w(s)) s^\mu}{\partial s} ds + O(t^{\mu-1}) + O(t^\mu),
\]

where \( \Pi = P_0 P_0^T \) and \( (a) \) is due to

\[
\lim_{t \to 0} \frac{a_t t^\mu}{t} = \lim_{t \to 0} a_t = 0.
\]

Then set \( s_i = E(w(t)) : \{ \Pi E_\Gamma(w(t)) \} \), we get

\[
\delta(t) = c t^\mu \Pi E^\mu_\Gamma(w(t)) + o(t^\mu) + O(t^{\mu-1}).
\]

To this end, by substituting (37) into (29), we have

\[
w_{\gamma, i}(t) \overset{d}{=} w_i^{SGD}(t) - \sqrt{\gamma} c t^\mu \Pi E^\mu_\Gamma(w(t)), \quad i = 1, 2, \ldots, |\mathcal{G}|,
\]
where \( \bar{s}_i = s_i + \frac{\sigma_i(\nu)}{c_i \sqrt{\pi}} + \frac{Q_i(\nu - 1)}{c_i \sqrt{\pi}} \) and hence \( \lim_{t \to \infty} |s_i - \bar{s}_i| = 0 \). Based on Assumption 6 we can further have
\[
w_{\gamma,i}(t) \doteq w_{i}^{SGD}(t) - c \sqrt{\gamma} t^\mu \bar{s}_i E(w_i^{SGD}(t)), \quad i = 1, 2, \ldots, |G|,
\]
hence we obtain (19). Based on Theorem 1, we further obtain (18). Then we complete the proof.

C. Proof of Theorem 3
Proof: To prove
\[
v_{\gamma}(t) \doteq w(t) + \sqrt{\gamma} c t^\mu E_G(w(t)) \nonumber
\]
\[-\sqrt{\gamma} c \sum_{k=1}^{K} \left\{ \Phi(t, T_k) \{E_G(w(T_k^\mu)) - E_G(w(T_k^-)) \} T_k \right\} \nonumber \]
\[-\sqrt{\gamma} c \int_0^t \Phi(t, s) \frac{\partial E_G(w(s))}{\partial s} ds \nonumber \]
\[+ \sqrt{\gamma} \int_0^t \Phi(t, s) \Sigma^{1/2}(w(s)) dB(s), \nonumber\]
we define the centered and scaled processes
\[V_{\gamma}(t) := \frac{v_{\gamma}(t) - w(t)}{\sqrt{\gamma}}, \tag{38}\]
then we need to prove
\[
V_{\gamma}(t) \doteq \sqrt{\gamma} t^\mu E_G(w(t)) \nonumber
\]
\[-c \sum_{k=1}^{K} \left\{ \Phi(t, T_k) \{E_G(w(T_k^\mu)) - E_G(w(T_k^-)) \} T_k \right\} \nonumber \]
\[-c \int_0^t \Phi(t, s) \frac{\partial E_G(w(s))}{\partial s} ds \nonumber \]
\[+ \int_0^t \Phi(t, s) \Sigma^{1/2}(w(s)) dB(s). \nonumber\]

By Theorem 13.13 in [31], \( V_{\gamma} \doteq V \) on \((T_k, T_k+1)\) for each \( k = 0, \ldots, K \) as \( \gamma \) is small, where \( V \) obeys the stochastic differential equation (SDE):
\[dV(t) = -H(w(t)) \nabla \zeta^*(V(t)) dt + \Sigma^{1/2}(w(t)) dB(t), \tag{39}\]
where the initial \( V(T_k) = V(T_k^-) \), \( B(t) \) is the \( d \)-dimensional standard Brownian motion, and \( \zeta^*(V(t)) \) is the Fenchel conjugate of \( \zeta(W(t)) \).

The function \( \zeta^*(\cdot) \) is defined as \( \zeta^*(w(t)) := \lim_{\gamma \to 0} \zeta_{\gamma}(w(t)) \) with \( \zeta_{\gamma}(w(t)) \) being the local Bregman divergence of \( \zeta_{\gamma}(w(t)) \) in (22) at \((v(t), w(t))\). In particular, we have
\[
\zeta_{\gamma}(w(t)) := \gamma^{-1} \left( \zeta_{\gamma}(w(t) + \sqrt{\gamma} u(t)) - \zeta_{\gamma}(w(t)) - \langle \sqrt{\gamma} u(t), w(t) \rangle \right) \nonumber
\]
\[= \frac{1}{2} \|w(t)\|^2 + \gamma^{-1} \left( \gamma \|w(t)\|^2 \right) \sum_{\mu} \left( \|w_{i}(t) + \sqrt{\gamma} u_{i}(t)\|^2 - \|w_{i}(t)\|^2 \right) \nonumber\]
where \( g(t/\gamma, \gamma) := \gamma^{1/2}(n \gamma)^{\mu/2} \) with \( c > 0 \) and \( n = \lfloor t/\gamma \rfloor \). Then we have
\[
\zeta_{\gamma}(w(t)) := \lim_{\gamma \to 0} \zeta_{\gamma}(w(t)) \nonumber
\]
\[= \frac{1}{2} \|w(t)\|^2 + c \gamma^{\mu} \sum_{i \in G} \|w_{i}(t)\|^2 \nonumber\]
\[= \frac{1}{2} \|w(t)\|^2 + c t^\mu \sum_{i \in G} \|w_{i}(t)\|^2 \nonumber\]

Hence, the derivative of \( \zeta^*(w(t)) \) satisfies
\[
\nabla \zeta^*(w(t)) = \arg \min_{w(t) \in R^d} \left\{ \zeta^*(w(t)) - w(t)^T v(t) \right\} \nonumber
\]
\[= \arg \min_{w(t) \in R^d} \left\{ \frac{1}{2} \|w(t)\|^2 + c t^\mu \sum_{i \in G} \|w_{i}(t)\|^2 - w(t)^T v(t) \right\}, \nonumber\]
where \( \nabla \zeta^*(w(t)) = \left[ \nabla \zeta^1_1(w(t))^T, \cdots, \nabla \zeta^1_{|G|}(w(t))^T \right]^T \)
and we have
\[
\nabla \zeta^*(w(t)) = \arg \min_{w_i(t)} \left\{ \frac{1}{2} \|w_i(t)\|^2 + c t^\mu \|w_i(t)\|^2 - w_i(t)^T v_i(t) \right\} \nonumber
\]
\[= \arg \min_{w_i(t)} \left\{ \frac{1}{2} \|w_i(t)\|^2 + c t^\mu \|w_i(t)\|^2 - w_i(t)^T v_i(t) \right\}, \nonumber\]
\[= \min_{w_i(t) \in R^d} \left\{ \frac{1}{2} \|w_i(t)\|^2 + c t^\mu \|w_i(t)\|^2 - w_i(t)^T v_i(t) \right\}. \nonumber\]

Next, based on Assumptions 3 and 4, the solution operator \( \Phi(t, s) \) of the inhomogeneous ODE system
\[dx(t) = -H(w(t)) x(t) dt, \quad x(t_0) = x_0 \]
uniquely exists, and the solution is \( x(t) = \Phi(t, t_0) x_0 \) by Theorem 5.1 of [54] and for \( 0 < s < m < t \),
\[
\Phi(t, s) = \Phi(t, m) \Phi(m, s), \tag{46}\]

Then (41) can be verified by (44) and Ito calculus for \( t \in (T_k, T_{k+1}) \) is given by
\[
V(t) = \Phi(t, t_0) V_0 + \int_{T_k}^{t} \Phi(t, s) H(w(s)) E_G(w(s)) c s^\mu ds \nonumber
\]
\[+ \int_{T_k}^{t} \Phi(t, s) \Sigma^{1/2}(w(s)) dB(s), \nonumber\]

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
since \( \frac{\partial \Phi(t, t_0) V_0}{\partial x} = -H(w(t)) \Phi(t, t_0) V_0 = -H(w(t)) V(t) \)
and we assume that the root of the coordinates in \( w(t) \) occur at time \( [T_k] \cap 0 = [T_k] \cap \infty \). By substituting with initial \( t_0 = T_k \) and \( V(T_k) = V(T_k) \), we have

\[
V(t) = \Phi(t, t_0) V(T_k) + \int_{t_0}^{t} \Phi(t, s) H(w(s)) E_G(w(s)) cs^u ds + \int_{t_0}^{t} \Phi(t, s) \Sigma L_2(w(s)) dB(s),
\]

(47)

the solution of (41). Note that \( V(T_0) = V(0) = V_\gamma(0) = 0 \) almost surely.

Set \( d \Delta_1(s) = H(w(s)) E_G(w(s)) cs^u ds \) and \( d \Delta_2(s) = \Sigma L_2(w(s)) dB(s) \). If \( t > T_k \), we have

\[
V(t) = \Phi(t, T_k) V(T_k) + \int_{T_k}^{t} \Phi(t, s) d \Delta_1(s) + \int_{T_k}^{t} \Phi(t, s) d \Delta_2(s)
\]

(48)

where \( (a) \) is by unfolding \( V(T_k) \) according to (47) with \( k = K - 1 \), \( (b) \) follows by (46) and \( V(T_0) = V(0) = 0 \).

We next analysis the first term in (48), which can be rewritten as

\[
\int_{0}^{t} \Phi(t, s) d \Delta_1(s)
\]

\[
= \int_{0}^{t} \Phi(t, s) H(w(s)) E_G(w(s)) cs^u ds
\]

(49)

where \( (a) \) follows by (45), and the first term in (49) can be further rewritten as

\[
\Phi(t, s) E_G(w(s)) cs^u \bigg|_{0}^{t} - \int_{0}^{t} \Phi(t, s) \frac{\partial E_G(w(s)) s^u}{\partial s} ds,
\]

(50)

where \( (a) \) is due to (42) and \( (b) \) is due to (43).

Combining (48), (49) and (50), we have

\[
V(t) = ct \mu E_G(w(t)) - c \sum_{k=1}^{K} \left\{ \Phi(t, T_k) \right\} E_G(w(T_k^+))
\]

\[
- E_G(w(T_k^+)) \int_{T_k}^{t} \Phi(t, s) \frac{\partial E_G(w(s)) s^u}{\partial s} ds
\]

\[
+ \int_{0}^{t} \Phi(t, s) \Sigma L_2(w(s)) dB(s).
\]

Based on (38), for \( t \in (T_k, T_{k+1}) \) we further have

\[
\nu_\gamma(\theta) = \mu \Phi(t, T_k) V(T_k) + \int_{T_k}^{t} \Phi(t, s) d \Delta_1(s) + \int_{T_k}^{t} \Phi(t, s) d \Delta_2(s)
\]

Then we complete the proof.

\[
\square
\]

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Adv. Neu. Inf. Process. Syst., vol. 25, pp. 1097–1105, 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[3] L. Deng and D. Yu, “Deep learning methods and applications,” Found. Trends Signal Process., vol. 7, nos. 3–4, pp. 197–387, 2014.

[4] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” 2018, arXiv:1809.11096.

[5] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556.

[6] M. Belskin, D. Hsu, S. Ma, and S. Mandal, “Reconciling modern machine-learning practice and the classical bias–variance trade-off,” Proc. Nat. Acad. Sci., vol. 116, no. 32, pp. 15849–15854, 2019.

[7] S-K. Chao, Z. Wang, Y. Xing, and G. Cheng, “Directional pruning of deep neural networks,” in Adv. Neu. Inf. Process. Syst., vol. 33, 2020, pp. 13986–13998.

[8] C. Xia, D. H. K. Tsang, and V. K. N. Lau, “Structured bayesian compression for deep neural networks based on the turbo-vb approach,” IEEE Trans. Signal Process., vol. 71, pp. 670–685, 2023.

[9] C. Lemaitre, A. Achkar, and P.-M. Judoin, “Structured pruning of neural networks with budget-aware regularization,” in Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit., 2019, pp. 9108–9116.

[10] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” in Proc. Int. Conf. Learn. Represent., 2018.

[11] S. Lin et al., “Towards optimal structured cnn pruning via generative adversarial learning,” in Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit., 2019, pp. 2790–2799.

[12] Y. Wang et al., “Pruning from scratch,” in Proc. AAAI Conf. Artif. Intell., vol. 34, no. 7, 2020, pp. 12273–12280.

[13] D. L. Donoho, “Compressed sensing,” IEEE Trans. Inf. Theory., vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[14] H. Yang, Z. Xu, I. King, and M. R. Lyu, “Online learning for group lasso,” in Proc. ICML, 2010.

[15] T. Ochiai, S. Matsuda, H. Watanabe, and S. Katagiri, “Automatic node selection for deep neural networks using group lasso regularization,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Piscataway, NJ, USA: IEEE Press, 2017, pp. 5485–5489.

[16] B. Hassibi and D. Stork, “Second order derivatives for network pruning: Optimal brain surgeon,” in Adv. Neural Inf. Process. Syst., vol. 5,1993.

[17] Y. LeCun, J. S. Denker, and S. A. Solla, “Optimal brain damage,” in Proc. Adv. Neu. Inf. Process. Syst., 1990, pp. 598–605.

[18] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning efficient convolutional networks through network slimming,” in Proc. IEEE Int. Conf. Comput. Vision, 2017, pp. 2376–2374.
Yanhui Geng (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in electronic engineering and information science from the University of Science and Technology of China (USTC), in 2002 and 2005, respectively, and the Ph.D. degree in electrical and electronic engineering from the University of Hong Kong (HKU) in 2009. He is the Head of Huawei Hong Kong Research Centre. Before that, he was the Head of Huawei Montreal Research Center from 2017 to 2020, and was a Principal Researcher with Huawei Noah’s Ark Laboratory Hong Kong, from 2013 to 2017. Before joining Huawei, he was a Postdoctoral Research Fellow with HKU, from 2009 to 2012, and was a Senior Scientific Engineer with the Hong Kong Applied Science and Technology Research Institute (ASTRI), from 2012 to 2013. His research interests include artificial intelligence, machine learning, information theory, wireless communication, and data-center networking. Dr. Geng has filed over 20 patents globally and he has over 30 publications on top journals and conferences. He has received numerous awards, including IEEE ICC 2010 Best Paper Award, Chinese Academy of Science President Award, Excellent Postgraduate of Anhui Province Award, and Excellent Postgraduate of USTC Award. He is a member of ACM.

Yinchuan Li (Member, IEEE) received the B.S. and Ph.D. degrees in electronic engineering from Beijing Institute of Technology (BIT), Beijing, China, in 2015 and 2020, respectively. From 2017 to 2019, he was a joint Ph.D. Student with the Department of Electrical Engineering, Columbia University, New York, NY, USA. From 2019 to 2020, he was a Senior Technical Consultant with Santé Ventures, Austin, TX, USA. Currently, he works with Noah’s Ark Laboratory, Huawei Technologies, Beijing, China, as a Principal Researcher. His research interests include generative models, machine learning, deep learning, and reinforcement learning. Dr. Li was the recipient of the Best Ph.D. Thesis Award of Chinese Institute of Electronics of 2022, and the IEEE International Conference on Signal, Information and Data Processing Excellent Paper Award of 2019. He has served as the Area Chair for ICLR 2023 Tiny Paper.