Only Train Once: A One-Shot Neural Network Training And Pruning Framework

Anonymous Author(s)
Affiliation
Address
email

Abstract

Structured pruning is a commonly used technique in deploying deep neural networks (DNNs) onto resource-constrained devices. However, the existing pruning methods are usually heuristic, task-specified, and require an extra fine-tuning procedure. To overcome these limitations, we propose a framework that compresses DNNs into slimmer architectures with competitive performances and significant FLOPs reductions by Only-Train-Once (OTO). OTO contains two keys: (i) we partition the parameters of DNNs into zero-invariant groups, enabling us to prune zero groups without affecting the output; and (ii) to promote zero groups, we then formulate a structured-sparsity optimization problem and propose a novel optimization algorithm, Half-Space Stochastic Projected Gradient (HSPG), to solve it, which outperforms the standard proximal methods on group sparsity exploration and maintains comparable convergence. To demonstrate the effectiveness of OTO, we train and compress full models simultaneously from scratch without fine-tuning for inference speedup and parameter reduction, and achieve state-of-the-art results on VGG16 for CIFAR10, ResNet50 for CIFAR10/ImageNet and Bert for SQuAD.

1 Introduction

Deep neural networks (DNNs) have been shown to be effective in various real applications [47, 27]. It is widely acknowledged that large-scale DNN models not only learn faster but also outperform their slimmer counterparts. However, such heavy models pose a great challenge to the deployment stage due to their resource-consuming nature. In addressing this issue, many model compression techniques [5, 11] are proposed in the past decade that aim at compressing those large and complex models into slimmer and simpler ones while suffering negligible loss in performance.

Pruning methods as one of the main categories of model compression, focus on identifying and pruning redundant structures via various mechanisms to achieve a slimmer architecture, and thus improve the interpretability of a DNN model [24, 11, 56]. For example, [31, 32] adopt fine-grained pruning via ℓ_1 or ℓ_2 regularization, which prune the small-weight connections based on some hard threshold. [35, 51, 54] measure the importance of filters to accelerate the networks by removing insignificant feature maps. [36, 7] utilize reinforcement learning agent to predict compression action.

Nevertheless, many of the existing pruning methods (i) often rely on criteria based on heuristics or empirical cues, *e.g.*, magnitude of a connection weight and importance score of a filter, to identify redundant parameters, which may cause divergence during optimization; (ii) thus require complex multi-stage pipelines that involve either a retraining or fine-tuning procedure to regain the accuracy during constructing a slimmer model, which is time-consuming; and (iii) are specific to certain architectures or applications, and are consequently less applicable to various downstream scenarios. Recently, there have been a few efforts [14, 52, 8] to directly train the network with sparsity inducing regularizers, which provide generality and convergence guarantee. However, these approaches focus

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

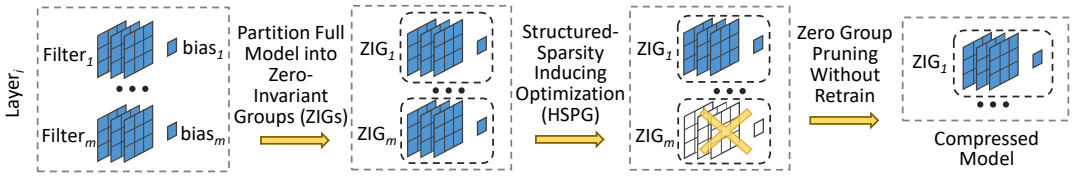


Figure 1: Overview of OTO. Without loss of generality, we illustrate OTO on a model with only vanilla convolutional layers, and for simplicity we only show Layer$_i$ with m 3D filters and their biases. The key to its success is twofold: (i) identify and partition the parameters of the model into zero-invariant groups (ZIGs); and (ii) solve the structured-sparsity regularization problem using HSPG. Finally, we obtain the compressed model by directly pruning the zero groups, *i.e.*, ZIG$_m$.

on either merely the individual sparsity of the parameters or the group sparsity of the filters, and thus cannot directly remove those zero components (still require subsequent fine-tuning) since the zeros are entangled with other commonly used components, *e.g.*, bias, batch normalization or skip connection. Furthermore, the optimization algorithms used in [14, 52] lack sufficient capability to explore (group) sparsity in DNNs effectively and require a post-processing step to yield exact zeros.

In this paper, we overcome the above limitations of existing pruning methods by proposing a *one-shot* neural network pruning framework, with which we are able to train a full heavy model from scratch only once, and obtain a slim architecture without fine-tuning while maintain high performance. As shown in Figure 1, the key to its success is twofold: (i) we identify and partition the parameters of DNNs into zero-invariant groups (ZIGs), enabling us to prune redundant structures according to zero groups without affecting the output of the network; and (ii) to promote zero groups, we formulate the pruning task as a structured-sparsity optimization problem and propose a novel optimization method, Half-Space Stochastic Projected Gradient (HSPG), to solve it, which outperforms the standard proximal methods on sparsity exploration and maintains comparable convergence. We highlight that both zero-invariant group partition and the novel optimization algorithm in promoting zero group lead to achieve one-shot neural network training and pruning regardless of its architecture.

Our main contributions are summarized as follows.

- **One-Shot Training and Pruning.** We propose OTO, a one-shot training and pruning framework that compresses a full neural network into a slimmer one with competitive performance by Only-Train-Once. OTO dramatically simplifies the complex multi-stage training pipelines of the existing pruning approaches, fits various architectures and applications, and hence is generic and efficient.
- **Zero-Invariant Group.** We define zero-invariant groups for neural networks. If a network is partitioned into ZIGs, it allows us to prune the zero groups without affecting the output, which results in one-shot pruning. Such property is applicable to various popular structures from plain fully connected layers to sophisticated ones such as residual blocks and multi-head attention.
- **Novel Structured-Sparsity Optimization Algorithm.** We propose Half-Space Stochastic Projected Gradient (HSPG), a method that solves structured-sparsity inducing regularization problem. We show and analyze the superiority of HSPG in promoting zero groups of networks than the standard proximal methods and the competitive objective convergence in practice. The fact that ZIG and HSPG are designed agnostic to networks makes OTO generic to various applications.
- **Experimental Results.** We train and compress full models simultaneously from scratch without fine-tuning for inference speedup and parameter reduction, and achieve state-of-the-art results on compression benchmark VGG for CIFAR10, ResNet50 for CIFAR10/ImageNet, Bert for SQuAD.

2 Related Work

Structured pruning focuses on identifying and pruning the redundant structures in a full model to achieve slimmer architectures for efficient model inference and storage [24, 31], where there have been numerous efforts dedicated. For CNN compression, the general procedure can be largely summarized as: (*i*) train a full model; (*ii*) identify and prune the redundant structures to build a slimmer model based on various criteria, including (structured) sparsity [52, 72, 14, 50, 87, 25, 87, 55], Bayesian pruning [86, 56, 53, 69], ranking kernels importance [48, 54, 40, 35, 51, 84], reinforcement learning [36, 7], adversarial robustness [65], scientific control [67], lottery ticket [21, 22, 63], joint quantization learning [68, 77], etc.; (*iii*) retrain or iteratively fine-tune the slimmer model to regain

the accuracy regression during pruning. These methods cannot avoid the extra and usually time-consuming fine-tuning step because the identified redundant structures, even parametrized with zeros, actually contribute to the model output, thereby additional fine-tuning step is an absolute necessity.

For pruning Bert [70], knowledge distillation [39] and LayerDropout [20] shorten Bert by reducing the number of layers directly. Other methods [28, 64, 29] build slimmer Berts in the manner of individual sparsity, but require specially designed data structure for storage and computing library to take advantage of sparse data [30, 10], and typically cannot achieve inference speedup against the highly optimized library [16] for dense model due to the discontiguous memory allocation [9].

The structured sparsity for weight pruning is the most relevant to the algorithm described in this paper. The existing structure learning works [52, 72, 14, 50, 87] have the respective disadvantages: *(i)* multiple trainings during the whole procedure since their group partition cannot isolate the impact of pruned structures to the model output; and *(ii)* heuristic post-processing to generate zero groups as the standard proximal methods [18, 74, 75, 12] and ADMM [84, 52, 4] defective on the sparsity exploration for deep learning [8], which may deteriorate the performance of the model significantly.

3 OTO

OTO is simple. In essence, OTO frames the network training and pruning as a structure learning problem. Given a full model $\mathcal{M}$, OTO trains and compresses it simultaneously from scratch *without* fine-tuning, and achieves significant reduction in both FLOPs and parameters. Particularly, as stated in Algorithm 1, the trainable parameters of $\mathcal{M}$ are firstly partitioned into a ZIG set $\mathcal{G}$ (Section 3.1). We then construct and solve a structured-sparsity inducing optimization problem (Section 3.2) by proposed stochastic optimizer (HSPG) to seek a highly group-sparse solution $\boldsymbol{x}^*_{\text{HSPG}}$ (Section 3.3). Lastly, we obtain a compressed model $\mathcal{M}^*$ by directly pruning these zero groups (Section 3.4).

Algorithm 1 Outline of OTO.

1: **Input:** Full model $\mathcal{M}$ (no need to be pretrained).
2: **Construct $\mathcal{G}$:** Partition the trainable parameters of $\mathcal{M}$ into a ZIG set $\mathcal{G}$.
3: **Train:** Train the model $\mathcal{M}$ using HSPG (Algorithm. 2) to obtain a group-sparse solution $\boldsymbol{x}^*_{\text{HSPG}}$.
4: **Prune:** Construct a slimmer model architecture $\mathcal{M}^*$ by directly pruning zero groups of $\boldsymbol{x}^*_{\text{HSPG}}$.
5: **Output:** Compressed model $\mathcal{M}^*$.

3.1 Zero-Invariant Group

The root cause of the existing methods having multi-stage training pipeline is that despite the pruned structure (*e.g.*, 3D filter) being zeros, its associated structures (*e.g.*, non-zero bias) still contribute to its corresponding output to the next layer (*e.g.*, feature map). As a result, the model accuracy regresses, hence an extra step of fine-tuning is necessary. OTO avoids the necessity by partitioning the parameters of DNNs into a set of so-called zero-invariant groups (ZIGs) $\mathcal{G}$ defined as follows.

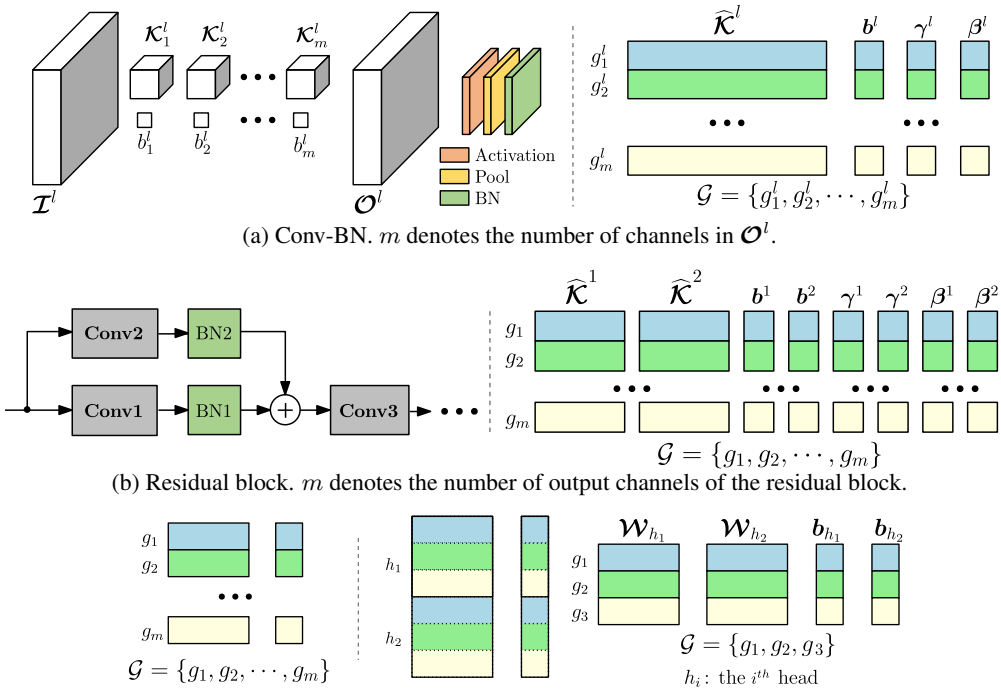
Definition 1 (Zero-Invariant Groups (ZIGs))**.** *For a layer-wise DNN, we partition its entire trainable parameters into disjoint groups $\mathcal{G} = \{g\}$. Then we call $\mathcal{G}$ zero-invariant groups (ZIGs) if each group $g \in \mathcal{G}$ is zero-invariant in the sense that all of the parameters in g being zeros results in its corresponding output to the next layer to be zeros as well.*

In effect, if and only if a DNN model is partitioned into a ZIG set $\mathcal{G}$ and one or more of its element g are parameterized by zeros, the entire corresponding structures contribute none to the model outputs and hence can be pruned directly. Such partition is applicable to various structures of DNN models. Without loss of generality, we define and describe ZIG partition for three most popular structures: *(i)* Conv-BN, *(ii)* Residual Block, and *(iii)* Fully Connected and Multi-Head Attention Layer.

ZIG of Conv-BN. Convolutional layer (Conv) followed by batch-normalization layer (BN) is extensively used in DNN models. Figure 2a shows the ZIG partition for Conv-BN. The 4D filter tensor $\boldsymbol{\kappa}^l$ is flattened into a filter matrix $\hat{\boldsymbol{\kappa}}^l$. During the forward pass, the input tensor $\boldsymbol{\mathcal{I}}^l$ is transformed into the output tensor $\boldsymbol{\mathcal{O}}^l$ of Conv and then into the input tensor of the $(l+1)^{th}$ layer $\boldsymbol{\mathcal{I}}^{l+1}$ by

$$\boldsymbol{\mathcal{O}}^l \leftarrow \boldsymbol{\mathcal{I}}^l \otimes \hat{\boldsymbol{\mathcal{K}}}^l + \boldsymbol{b}^l,\ \boldsymbol{\mathcal{I}}^{l+1} \leftarrow \frac{a(\boldsymbol{\mathcal{O}}^l) - \boldsymbol{\mu}^l}{\boldsymbol{\sigma}^l} \odot \boldsymbol{\gamma}^l + \boldsymbol{\beta}^l, \tag{1}$$

where denoted by $\otimes$ the convolutional operation, $\odot$ the element-wise multiplication and $a(\cdot)$ the activation function. BN is parameterized by mean $\boldsymbol{\mu}^l$, standard deviation $\boldsymbol{\sigma}^l$, weight $\boldsymbol{\gamma}^l$ and bias $\boldsymbol{\beta}^l$



(a) Conv-BN. m denotes the number of channels in $\mathcal{O}^l$.

(b) Residual block. m denotes the number of output channels of the residual block.

(c) Fully connected layer (Left). Multi-head attention layer (Right). m denotes the length of output vector.

Figure 2: Zero-invariant group partition for three popular structures.

respectively. The activation function needs to be zero-invariant, *i.e.*, $a(\mathbf{0}) = \mathbf{0}$, where most instances satisfy, *e.g.*, ReLU [23], PReLU [33], GELU [38] and LeakyReLU [76]. Hence, each row of the flattened filter matrix $\hat{\mathcal{K}}^l$ and its bias $\boldsymbol{b}^l$ belong to one ZIG because they being zeros results in their corresponding channel of $\mathcal{O}^l$ (*i.e.*, feature map) to be zeros as well. Subsequently, $\boldsymbol{\gamma}^l$ and $\boldsymbol{\beta}^l$ of this corresponding channel in BN are also included into this ZIG to avoid the value shift (zero to non-zero) during normalization. Note that grouping these four sets of parameters channel-wisely makes Conv-BN zero-invariant regardless of the value of $\boldsymbol{\mu}^l$ and $\boldsymbol{\sigma}^l$, and hence they are excluded from the ZIG. For illustration, each ZIG is highlighted in the same color (*e.g.*, g_1^l in blue).

ZIG of Residual Block. The residual block adds another layer of challenge because its output tensor is the summation of the outputs of two Conv-BNs. Figure 2b shows the ZIG partition for the residual block. As illustrated, before propagated to Conv3, the outputs of Conv1-BN1 and Conv2-BN2 are summarized and hence share the same dimension. As such, to make residual block zero-invariant, we group the four sets of parameters channel-wisely of both Conv1-BN1 and Conv2-BN2 into ZIGs, *i.e.*, each row of $\hat{\mathcal{K}}^1$, $\boldsymbol{b}^1$, $\boldsymbol{\gamma}^1$, $\boldsymbol{\beta}^1$ of Conv1-BN1 and each row of $\hat{\mathcal{K}}^2$, $\boldsymbol{b}^2$, $\boldsymbol{\gamma}^2$, $\boldsymbol{\beta}^2$ of Conv2-BN2. In Appendix A.1, we describe the zero-invariant group partition of ResNet50 in greater detail.

ZIG of Fully Connected and Multi-Head Attention Layer. Figure 2c shows the ZIG partition for fully connected and multi-head attention layer. Particularly, we partition each row of weight matrix and its associated bias into a ZIG, and therefore any input element is turned to zero if that ZIG is parameterized with zeros, making the fully connected layer zero-invariant. Multi-head attention layer is the key building block of the transformer architectures [70]. Its trainable parameters contain a weight matrix and bias vector, consisting of the sub-matrix and sub-vector of each head (we use two heads as an example). We form ZIG by grouping each row of every sub-matrix and sub-vector, *i.e.*, each row of $\boldsymbol{\mathcal{W}}_{h_1}$, $\boldsymbol{b}_{h_1}$, $\boldsymbol{\mathcal{W}}_{h_2}$ and $\boldsymbol{b}_{h_2}$ of h_1 and h_2, respectively.

3.2 Structured-Sparsity Regularization

We now formulate a structured-sparsity regularization problem over the ZIG set $\mathcal{G}$ for the trainable parameters of the full model $\mathcal{M}$ as follows

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}}\ \psi(\boldsymbol{x}) := f(\boldsymbol{x}) + \lambda r(\boldsymbol{x}),\ r(\boldsymbol{x}) := \sum_{g\in\mathcal{G}} \|[\boldsymbol{x}]_g\|\,, \tag{2}$$

where $\lambda > 0$ is a weighting coefficient, $f(\boldsymbol{x})$ is a task-specific loss function, and $r(\boldsymbol{x})$ is an augmented structured-sparsity inducing regularization term encoding the topological structure of $\mathcal{M}$ over $\mathcal{G}$. A larger λ typically results in a higher group sparsity while sacrifices more on the bias of model estimation. We aim at computing a local optimum to achieve both low loss and high group sparsity.

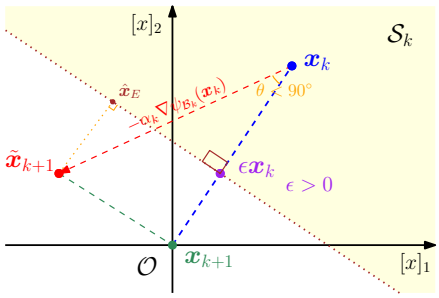
To induce group sparsity onto the solution of (2), there exist several candidates for $r(\boldsymbol{x})$, including mixed ℓ_1/ℓ_p norm ($p > 1$) [1, 19] and group Minmax Concave Penalty (MCP) [80]. Among these candidates, the mixed ℓ_1/ℓ_2 norm as defined in (2) is arguably the most popular choice in classical machine learning applications [1, 78], where $\|\cdot\|$ is the ℓ_2-norm, and each component $g \in \mathcal{G}$ indexes a group of variables. In this paper, we will demonstrate the effectiveness of OTO by selecting $r(\boldsymbol{x})$ as the mixed ℓ_1/ℓ_2 norm. We highlight OTO is applicable for other group sparsity regularizers as well.

3.3 Half-Space Stochastic Projected Gradient (HSPG)

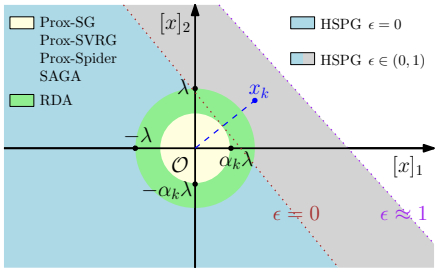
To solve the non-smooth regularization problem as (2) in deep learning applications, the standard proximal method and the ADMM lack capability to effectively identify group sparsity; see the discussions later in this Section. Therefore, we propose a novel stochastic optimization algorithm so-called Half-Space Stochastic Projected Gradient (HSPG) to enhance the group sparsity exploration more effectively than the classical methods while maintain a similar convergence property.

Outline. We state the outline of HSPG in Algorithm 2. It contains two stages: Initialization Stage and Group-Sparsity Stage. The first Initialization Stage employs Stochastic Gradient Descent (SGD) step to search for a good but usually non-sparse solution estimate. Then the second stage proceeds Half-Space step started with the non-sparse iterate to effectively exploit the group sparsity within a sequence of reduced spaces and converges to the group-sparse solutions.

Initialization Stage. The Initialization Stage performs the vanilla SGD to find a good initial point for the subsequent Group-Sparsity Stage. At k^{th} iteration, a stochastic gradient of f, *e.g.*, based on a mini-batch, is generated denoted as $\nabla\tilde{f}$. Since the group sparsity inducing regularizer $r(\boldsymbol{x})$ in the form as (2) is non-smooth, we select a subgradient $\zeta(\boldsymbol{x}_k)$ from its subdifferential $\partial r(\boldsymbol{x}_k)$ to form a stochastic subgradient of $\psi(\boldsymbol{x}_k)$ as $\nu(\boldsymbol{x}_k) := \nabla\tilde{f}(\boldsymbol{x}_k) + \lambda\zeta(\boldsymbol{x}_k)$. We then compute the next iterate as $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k - \alpha_k\nu(\boldsymbol{x}_k)$ by subgradient descent update.



(a) Half-Space Projection



(b) Projection Region For Mixed ℓ_1/ℓ_2 Regularization

Figure 3: Illustration of Half-Space Step with projection in (6), where $\mathcal{G} = \{\{1, 2\}\}$.

Group-Sparsity Stage. The Group-Sparsity Stage is designed to effectively determine the groups of zero variables and capitalize convergence characteristic, which is in sharp contrast to other heuristic aggressive weight pruning methods that typically lack theoretical guarantees [49, 54]. The intuition of Half-Space Step is to project $[\boldsymbol{x}_k]_g$ to zero only if $-[\boldsymbol{x}_k]_g$ serves as a descent step to $\psi(\boldsymbol{x}_k)$, *i.e.*, $-[\boldsymbol{x}_k]_g^\top[\nabla\psi(\boldsymbol{x}_k))]_g < 0$, hence updating $[\boldsymbol{x}_{k+1}]_g \leftarrow [\boldsymbol{x}_k]_g - [\boldsymbol{x}_k]_g = 0$ still results in some progress to the optimality. In particular, we first define the following index sets for any $\boldsymbol{x} \in \mathbb{R}^n$:

$$\mathcal{I}^0(\boldsymbol{x}) := \{g : g \in \mathcal{G}, [\boldsymbol{x}]_g = 0\} \ \text{ and } \ \mathcal{I}^{\neq 0}(\boldsymbol{x}) := \{g : g \in \mathcal{G}, [\boldsymbol{x}]_g \neq 0\}, \tag{3}$$

where $\mathcal{I}^0(\boldsymbol{x})$ represents the indices of groups of zero variables at $\boldsymbol{x}$, and $\mathcal{I}^{\neq 0}(\boldsymbol{x})$ indexes the groups of nonzero variables at $\boldsymbol{x}$. To proceed, we further define an artificial set that $\boldsymbol{x}$ lies in:

$$\mathcal{S}(\boldsymbol{x}) := \{\boldsymbol{0}\} \bigcup \{\boldsymbol{z} \in \mathbb{R}^n : [\boldsymbol{z}]_g = \boldsymbol{0} \text{ if } g \in \mathcal{I}^0(\boldsymbol{x}), \text{and } [\boldsymbol{z}]_g^\top[\boldsymbol{x}]_g \geq \epsilon\,\|[\boldsymbol{x}]_g\|^2 \text{ if } g \in \mathcal{I}^{\neq 0}(\boldsymbol{x})\}, \tag{4}$$

which consists of half-spaces and the origin. Here the parameter $\epsilon \geq 0$ controls how aggressively we promote group sparsity, and is typically fixed as zero in practice. Hence, $\boldsymbol{x} \in \mathcal{S}_k := \mathcal{S}(\boldsymbol{x}_k)$ only if: *(i)* $[\boldsymbol{x}]_g$ lies in the upper half-space for all $g \in \mathcal{I}^{\neq 0}(\boldsymbol{x}_k)$ for some prescribed $\epsilon \in [0, 1)$ as shown in Figure 3a; and *(ii)* $[\boldsymbol{x}]_g$ equals to zero for all $g \in \mathcal{I}^{0}(\boldsymbol{x}_k)$. Intuitively, $\mathcal{S}_k$ establishes the region where important structures inhabit, thereby redundant structures vanish if falling outside.

Ideally, the Initialization Stage has produced reasonably well but typically non-sparse iterate $\boldsymbol{x}_k$ nearby a group-sparse solution $\boldsymbol{x}^*$ of problem (2), , *i.e.*, the optimal distance $\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$ is sufficiently small. As seen in Appendix B, it further indicates that the group-sparse optimal solution $\boldsymbol{x}^*$ inhabits $\mathcal{S}_k$, and $\mathcal{S}_k$ has already covered the group-support of $\boldsymbol{x}^*$, *i.e.*, $\mathcal{I}^{\neq 0}(\boldsymbol{x}^*) \subseteq \mathcal{I}^{\neq 0}(\boldsymbol{x}_k)$. Our goal now becomes minimizing $\psi(\boldsymbol{x})$ over $\mathcal{S}_k$ to identify the remaining zero groups, *i.e.*, $\mathcal{I}^{0}(\boldsymbol{x}^*)/\mathcal{I}^{0}(\boldsymbol{x}_k)$, which is formulated as the following problem:

$$\underset{\boldsymbol{x} \in \mathcal{S}_k}{\text{minimize}}\ \psi(\boldsymbol{x}) = f(\boldsymbol{x}) + \lambda r(\boldsymbol{x}). \quad (5)$$

Algorithm 2 Outline of HSPG for solving (2).

1: **Input:** $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\epsilon \in [0, 1)$, and $N_{\mathcal{P}} \in \mathbb{Z}^+$.
2: **Output:** a group-sparse solution $\boldsymbol{x}^*_{\text{HSPG}}$ from $\{\boldsymbol{x}_k\}$.
3: **for** $k = 0, 1, 2, \ldots$ **do**
4: Compute a stochastic subgradient $\nu(\boldsymbol{x}_k)$ of $\psi(\boldsymbol{x}_k)$.
5: **if** $k < N_{\mathcal{P}}$ **then**
6: ***Subgradient Descent Update:***
7: Set $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \alpha_k \nu(\boldsymbol{x}_k)$.
8: **else**
9: ***Half-Space Update:***
10: Set a trial iterate $\tilde{\boldsymbol{x}}_{k+1}$ as
$$[\tilde{\boldsymbol{x}}_{k+1}]_{\mathcal{I}^{\neq 0}(\boldsymbol{x}_k)} \leftarrow [\boldsymbol{x}_k - \alpha_k \nu(\boldsymbol{x}_k)]_{\mathcal{I}^{\neq 0}(\boldsymbol{x}_k)}$$
$$[\tilde{\boldsymbol{x}}_{k+1}]_{\mathcal{I}^{0}(\boldsymbol{x}_k)} \leftarrow \boldsymbol{0}.$$
11: **for** each group g in $\mathcal{G}$ **do**
12: $[\boldsymbol{x}_{k+1}]_g \leftarrow [\text{Proj}^{HS}_{\mathcal{S}_k}(\tilde{\boldsymbol{x}}_{k+1})]_g$.
13: Update α_{k+1}.

The next iterate $\boldsymbol{x}_{k+1}$ is computed as an solution estimate of problem (5). Particularly, in Algorithm 2, $[\boldsymbol{x}_{k+1}]_{\mathcal{I}^{0}(\boldsymbol{x}_k)} \equiv \boldsymbol{0}$ will not be updated, and only the entries in $\mathcal{I}^{\neq 0}(\boldsymbol{x}_k)$ are free to move. Hence $\psi(\boldsymbol{x})$ is smooth on $\mathcal{S}_k$, and (5) is a reduced space optimization problem. A standard way to solve problem (5) would be the stochastic gradient descent equipped with Euclidean projection [58]. However, such a projected method rarely produces zero (group) variables, as the dense Euclidean projected point $\hat{\boldsymbol{x}}_E \neq \boldsymbol{0}$ illustrated in Figure 3a. To address, we introduce a novel half-space projection operator to effectively project an entire group of variables to zeros.

As line 4 and 9-12 in Algorithm 2, we first approximate the (sub)gradient of ψ on the free variables by $[\nu(\boldsymbol{x}_k)]_{\mathcal{I}^{\neq 0}(\boldsymbol{x}_k)}$, then employ gradient descent over $\mathcal{I}^{\neq 0}(\boldsymbol{x}_k)$ to compute a trial point $\widetilde{\boldsymbol{x}}_{k+1}$ which is passed into a fresh half-space projection operator $\text{Proj}^{HS}_{\mathcal{S}_k}(\cdot)$ defined as

$$\left[\text{Proj}^{HS}_{\mathcal{S}_k}(\boldsymbol{z})\right]_g := \begin{cases} 0 & \text{if } [\boldsymbol{z}]_g^\top [\boldsymbol{x}_k]_g < \epsilon \left\|[\boldsymbol{x}_k]_g\right\|^2, \\ [\boldsymbol{z}]_g & \text{otherwise.} \end{cases} \quad (6)$$

The above projector of form (6) is not the standard one in Euclidean sense[1], and it has two advantages: *(i)* the actual search direction $\boldsymbol{d}_k := (\text{Proj}^{HS}_{\mathcal{S}_k}(\tilde{\boldsymbol{x}}_{k+1}) - \boldsymbol{x}_k)/\alpha_k$ performs as a descent direction to $\psi(\boldsymbol{x}_k)$, *i.e.*, $[\boldsymbol{d}_k]_g^\top [\nu(\boldsymbol{x}_k))]_g < 0$ as $\theta < 90^\circ$ in Figure 3a, hence the progress to the optimum is made via the sufficient decrease property drawn as Lemma 1 in Appendix B; then *(ii)* it effectively projects entire groups of variables to zero if the inner product of corresponding entries is sufficiently small. In contrast, the Euclidean projection operator is far away effective to promote group sparsity.

Superiority of HSPG on Group Sparsity Identification. We now intuitively illustrate the strength of HSPG on group sparsity exploration. In fact, the half-space projection (6) is a more effective sparsity promotion mechanism compared to the standard proximal methods. Particularly, it benefits from a much larger projection region to map a reference point $\hat{\boldsymbol{x}}_{k+1} := \boldsymbol{x}_k - \alpha_k \nabla \tilde{f}(\boldsymbol{x}_k)$ or its variants to zero. As the 2D case described in Figure 3b, the projection regions of the state-of-the-art Prox-SG [18], Prox-SVRG [75], Prox-Spider [81] and SAGA [12] for (2) are ℓ_2-balls with radius as $\alpha_k \lambda$. In deep learning applications, the step size α_k is usually selected around 10^{-3} to 10^{-4} or even smaller for convergence. Together with the common setting of $\lambda \ll 1$, their projection regions would vanish rapidly, resulting in the difficulties to produce group sparsity. As a sharp contrast, even though $\alpha_k \lambda$ is near zero, the projection region of HSPG $\{\boldsymbol{x} : \boldsymbol{x}_k^\top \boldsymbol{x} < (\alpha_k \lambda + \epsilon \|\boldsymbol{x}_k\|) \|\boldsymbol{x}_k\|\}$ (seen in Appendix B) is still an open half-space which contains those ℓ_2 balls as well as RDA [74]'s if ϵ is

[1]Note that when $r(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ where each $g \in \mathcal{G}$ is singleton, then $\mathcal{S}_k$ becomes an orthant face [8].

Table 1: VGG16 and VGG16-BN for CIFAR10. Convolutional layers are in bold.

Method	BN	Architecture	FLOPs	# of Params	Top-1 Acc.
Baseline	✗	**64-64-128-128-256-256-256-512-512-512-512-512-512**-512-512	100%	100%	91.6%
SBP [56]	✗	**47-50-91-115-227-160-50-72-51-12-34-39-20**-20-272	31.1%	5.9%	**91.0%**
BC [53]	✗	**51-62-125-128-228-129-38-13-9-6-5-6-6**-6-20	38.5%	5.4%	**91.0%**
RBC [86]	✗	**43-62-120-120-182-113-40-12-20-11-6-9-10**-10-22	32.3%	3.9%	90.5%
RBP [86]	✗	**50-63-123-108-104-57-23-14-9-8-6-7-11**-11-12	28.6%	2.6%	**91.0%**
OTO	✗	**3-21-45-82-110-109-68-37-13-9-7-3-5-8**-170-344	**16.3%**	**2.5%**	**91.0%**
Baseline	✓	**64-64-128-128-256-256-256-512-512-512-512-512-512**-512-512	100%	100%	93.2%
EC [49]	✓	**32-64-128-128-256-256-256-256-256-256-256-256-256**-512-512	65.8%	37.0%	93.1%
Hinge [50]	✓	–	60.9%	20.0%	**93.6%**
OTO	✓	**3-22-56-93-123-182-125-95-45-27-21-10-13-19**-244-392	**26.8%**	**5.5%**	93.3%

large enough. Conversely, vanilla ADMM alone lacks the mechanism to project a group of variables to zero, unless equips with extra post-processing step [84, 52]. In Appendix B, we further reveal that HSPG recovers group sparsity under a better radius assumption than the proximal operator in Theorem 2 and still maintains the convergence to the optimality as drawn in Theorem 1. Moreover, we numerically demonstrate the superiority of HSPG in the sense of optimization in Appendix C.

3.4 One-Shot Pruning

The group-sparse solution $\boldsymbol{x}^*_{\text{HSPG}}$ over ZIGs to the full model $\mathcal{M}$ is leveraged to construct the slimmer model $\mathcal{M}^*$. Particularly, we prune the redundant structures identified as zero groups $\mathcal{I}^0$ and retain non-zero groups $\mathcal{I}^{\neq 0}$ in $\boldsymbol{x}^*_{\text{HSPG}}$. Because the parameters of full model are partitioned into ZIGs, the pruned structures contribute none to the model output. Therefore, given the same input, the slimmer model $\mathcal{M}^*$ computes the identical output as the full model $\mathcal{M}$ parameterized with $\boldsymbol{x}^*_{\text{HSPG}}$.

4 Experiment

In this section, we numerically demonstrate the effectiveness of OTO by one-shot training and pruning without fine-tuning on several benchmark compression tasks for CNNs, *i.e.*, VGG16 [66] for CIFAR10 [45] and ResNet50 [34] for CIFAR10 [45] and ImagetNet (ILSVRC2012) [15]. We also verify the scalibility of OTO onto Bert [70] evaluated on SQuAD [60]. All datasets are free to academic usage and do not contain personally identifiable information or offensive content. CIFAR10 is under the MIT license, consisting of 50,000 training and 10,000 test images from 10 classes. ImagetNet is a large-scale dataset without license and contains about 1.2 million and 50,000 images in training and validation sets from 1,000 classes. SQuAD is under the CC BY-SA 4.0 license with about 100,000 question/answer pairs splitted into train/dev/test sets as (80/10/10%). We conduct all experiments on a Nvidia RTX8000 GPU and provide implementation details in Appendix A.

4.1 Deep Convolutional Neural Network

The results on CNN experiments are summarized in Table 1, 2 and 3. In particular, we compare OTO to its state-of-the-art counterparts by Top-1/5 accuracy, remaining FLOPs and parameters against the corresponding baseline (full model). We report the numbers of other methods based on the corresponding literature and leave as '-' if not reported. The best pruning results are marked as bold.

VGG16 for CIFAR10. We consider the standard VGG16 and the version with batch normalization layer after each convolutional layer, referred to as VGG16-BN. OTO partitions the parameters into ZIGs following Section 3.1, then trains and prunes the model via HSPG, and finally constructs the slimmer model without fine-tuning. For VGG16, as shown in Table 1, the pruned architecture of OTO indicates that OTO identifies similar redundancy of the intermediate and late convolutional layers compared to other methods, but significantly more of the early convolutional layers. As a result, OTO achieves 83.7% $(1 - 16.3\%)$ FLOPs reduction and 97.5% $(1 - 2.5\%)$ parameter reduction with the best Top-1 accuracy, which outperforms other state-of-the-arts significantly. For VGG16-BN, among all, OTO reduces FLOPs and parameters to the lowest 26.8% and 5.5%, respectively. EC [49] and Hinge [50] achieve the same level of Top-1 accuracy as OTO, but are substantially outperformed when it comes to FLOPs and parameter reduction.

ResNet50 for CIFAR10. We also compare OTO with two state-of-the-art automatic neural network compression frameworks, *i.e.*, AMC [36] and ANNC [77]. AMC trains a reinforcement learning agent to predict compression action for each layer environment. ANNC jointly proceeds pruning and quantization within energy constraint. We conduct OTO on their shared experiment, *i.e.*, ResNet50 on CIFAR10. ResNet50 includes both the standard convolutional layers and the layers with residual connections, which are partitioned into ZIGs following Section 3.1. We report the results in Table 2. Based on the results, all three methods achieve competitive validation accuracies higher than the baseline reported in [36]. OTO outperforms AMC and ANNC without quantization by using only 12.8% FLOPs and 8.8% parameters. Note that no FLOPs reduction is reported in [36] and [77]. Finally, we highlight that OTO is flexible to incorporate quantization as the two techniques are complementary and will leave to future work.

Table 2: ResNet50 for CIFAR10.

Method	FLOPs	# of Params	Top-1 Acc.
Baseline	100%	100%	93.5%
AMC [36]	–	60.0%	93.6%
ANNC [77]	–	50.0%	**95.0%**
OTO	**12.8%**	**8.8%**	94.4%

ResNet50 for ImageNet. We now evaluate OTO on ResNet50 for ImageNet. As shown in Table 3, OTO prunes $64.5\%(1 - 35.5\%)$ parameters to achieve $65.5\%(1-34.5\%)$ FLOPs reduction with only $1.4\%/0.8\%$ Top-1/5 accuracy regression compared to the baseline. OTO consistently outperforms the counterparts especially on the FLOPs reduction and the parameter reduction. We note that Hinge [50] prunes CNNs via structured-sparsity optimization by employing standard stochastic proximal gradient method. It requires several trainings including fine-tuning the pruned model, because it partitions the parameters into non-ZIGs and relies on an empirical truncation mechanism to generate zero groups due to the weakness of proximal operator in deep learning applications [8]. In contrast, OTO only trains and prunes the full model from scratch once and obtains better pruning results. The comparison between OTO and Hinge stand as evidence of the superiority of OTO due to ZIGs and HSPG.

Table 3: ResNet50 for ImageNet.

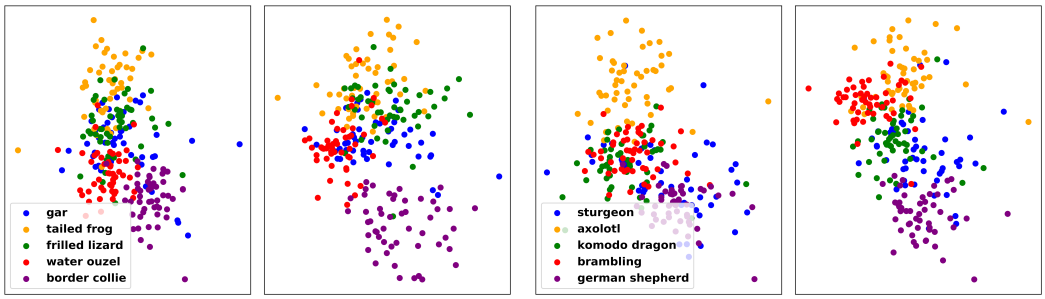
Method	FLOPs	# of Params	Top-1 Acc.	Top-5 Acc.
Baseline	100%	100%	76.1%	92.9%
DDS-26 [41]	57.0%	61.2%	71.8%	91.9%
CP [37]	66.7%	–	72.3%	90.8%
ThiNet-50 [43]	44.2%	48.3%	71.0%	90.0%
RBP [86]	43.5%	48.0%	71.1%	90.0%
RRBP [86]	45.4%	–	73.0%	91.0%
SFP [35]	41.8%	–	74.6%	**92.1%**
Hinge [50]	46.6%	–	**74.7%**	–
OTO	**34.5%**	**35.5%**	**74.7%**	**92.1%**

Representation of Deep Features of ImageNet. It is widely acknowledged that deep neural architectures could be treated as non-linear feature representation extractors. Therefore, we further study the feature representation extracted by OTO to demonstrate its generalizability to other visual applications besides image classification. Figure 4 shows the clustering results of ImageNet validation images using the deep feature extracted by both the baseline ResNet50 and the pruned ResNet50 by OTO. Specifically, we extract the deep features over the validation samples in ImageNet, *i.e.*, the tensors fed into the fully connected layer, and project them onto a 2-dimensional space via PCA [44]. For illustration, following the hierarchy of ImageNet [3], two sets of five classes are randomly selected[2]. We observe that the deep features of the pruned ResNet50 by OTO remain structured in the sense that distinct classes are well separated from each other. Over all 1000-class ImageNet validation images, OTO achieves 48.2% clustering accuracy compared to 42.5% of the baseline ResNet50 using k-means. Both observations indicate that with only 35.5% parameters and 34.5% FLOPs, the pruned ResNet50 is still able to extract highly discriminative deep features. We argue that during model compression, OTO not only achieves parameter and FLOPs reduction, but also preserves the ability of capturing perceptual properties [83]. This is especially important in training and compressing models for many vision tasks, *e.g.*, object detection [61, 62], frame interpolation [2, 17, 57] and video synthesis [71, 46]. We leave the application of OTO to broader tasks to future work.

4.2 Large-Scale Transformer

We show the scalability of OTO by pruning the large-scale transformer Bert [70], evaluated on SQuAD, a question-answering benchmark [60]. Bert mainly includes embedding layers, fully connected layers and multi-head attention layers. The fully connected layers and the multi-head

[2] Each selected class belongs to a disjoint upper category.



(a) (b)

Figure 4: Clustering results of ImageNet validation images using deep features extracted by full ResNet50 (left of a and b) and pruned ResetNet50 by OTO (right of a and b). The points are visualized by projecting deep features onto a two-dimensional space via PCA.

attention layers are partitioned into ZIGs following Section 3.1. For fair comparisons, we follow the prior Bert compression works [14, 64] and do not prune the embedding layers.

To the best of our knowledge, OTO is the first work that compresses Bert by exploring group sparsity on individual layers and achieves significant parameter reduction and inference speedup[3]. In contrast, the existing works [28, 64, 29] prune individual parameters instead, *i.e.*, the generated sparsity is not structured. Hence, the computed models typically do not have inference speedup [64], unless are executed by specialized hardware and sparse computing library [30, 10]. As shown in Table 4, under different group sparsity upper bound constraints, OTO reduces 9% to 60% parameters and achieves up to $1.8\times$ inference speedup based on the average model execution time[4]. In comparison, despite that the pruned model contains 10% parameters, MaP and MvP [64] do not have any inference speedup. On the other hand, the structured sparsity on Bert is studied in [14] (referred to as ProxSSI), where an adaptive proximal method is proposed to yield group-sparse solution. Nonetheless, ProxSSI optimizes over non-ZIGs and relies on proximal operator to identify group sparsity. Therefore, the groups even parameterized with zeros have to be retained in the model rather than pruned. As a consequence, ProxSSI is not competitive to OTO on parameter reduction, and there is no reported inference speedup. Note that all the pruning methods achieve comparable exact match rate and F1-score.

Table 4: Pruning Bert on SQuAD

Method	# of Params	Exact	F1-score	SpeedUp
Baseline	100%	81.0%	88.3%	$1\times$
MaP [64]	**10.0%**	67.7%	78.5%	$1\times$*
MvP [64]	**10.0%**	71.9%	81.7%	$1\times$*
ProxSSI [14]	83.4%†	72.3%	82.0%	$1\times$
OTO	91.0%	**75.0%**	**84.1%**	$1.1\times$
OTO	76.2%	72.3%	82.1%	$1.2\times$
OTO	66.7%	71.9%	82.0%	$1.3\times$
OTO	53.3%	71.4%	81.5%	$1.5\times$
OTO	40.0%	70.9%	81.1%	$\mathbf{1.8\times}$

* Based on the statement in the official git repository of [64].
† Approximate value based on the group sparsity reported in [14].

5 Conclusion And Future Work

We propose OTO, a one-shot deep neural networks (DNNs) training and pruning framework, that compresses full DNNs into slimmer architectures with competitive performances and significant FLOPs and parameter reduction without fine-tuning. OTO contains two fundamentals: *(i)* partitions the trainable parameters of DNNs into zero-invariant groups (ZIGs), thereby pruning zero groups does not affect the model output, and *(ii)* trains by a novel optimizer, Half-Space Stochastic Projected Gradient (HSPG), which outperforms proximal methods on group sparsity exploration and maintains comparable convergence. We numerically demonstrate OTO on benchmark experiments, *i.e.*, VGG16 for CIFAR10, ResNet50 for CIFAR10/ImageNet and Bert for SQuAD, and achieve state-of-the-art pruning results. We leave incorporating quantization and applying OTO to other tasks to future work.

[3] Knowledge distillation [39] and LayerDropout [20] compresses Bert by pruning layers in their entirety.
[4] Run by OnnxRuntime [16].

References

[1] Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 2012.

[2] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3703–3712, 2019.

[3] Mike Bostock. Imagenet hierarchy. https://observablehq.com/@mbostock/imagenet-hierarchy.

[4] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[5] Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[6] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[7] Jianda Chen, Shangyu Chen, and Sinno Jialin Pan. Storage efficient and dynamic flexible runtime channel pruning via deep reinforcement learning. 2019.

[8] Tianyi Chen, Tianyu Ding, Bo Ji, Guanyi Wang, Yixin Shi, Sheng Yi, Xiao Tu, and Zhihui Zhu. Orthant based proximal stochastic gradient method for ell_1-regularized optimization. *arXiv preprint arXiv:2004.03639*, 2020.

[9] Tianyi Chen, Yixin Shi, and Sheng Yi. Spatially sparse convolutional neural networks for inking applications, Sept. 17 2020. US Patent App. 16/355,702.

[10] Xuhao Chen. Escoin: Efficient sparse convolutional neural network inference on gpus. *arXiv preprint arXiv:1802.10280*, 2018.

[11] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[12] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

[13] Aaron Defazio and Léon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. In *Advances in Neural Information Processing Systems*, 2019.

[14] Tristan Deleu and Yoshua Bengio. Structured sparsity inducing adaptive optimizers for deep learning. *arXiv preprint arXiv:2102.03869*, 2021.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[16] ONNX Runtime developers. Onnx runtime. https://onnxruntime.ai/, 2021.

[17] Tianyu Ding, Luming Liang, Zhihui Zhu, and Ilya Zharkov. Cdfi: Compression-driven network design for frame interpolation. *arXiv preprint arXiv:2103.10559*, 2021.

[18] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10(Dec):2899–2934, 2009.

[19] Marwa El Halabi, Francis Bach, and Volkan Cevher. Combinatorial penalties: Which structures are preserved by convex relaxations? In *International Conference on Artificial Intelligence and Statistics*, pages 1551–1560. PMLR, 2018.

[20] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

[21] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[22] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

[23] Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.

[24] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[25] Shang-Hua Gao, Yong-Qiang Tan, Ming-Ming Cheng, Chengze Lu, Yunpeng Chen, and Shuicheng Yan. Highly efficient salient object detection with 100k parameters. In *European Conference on Computer Vision*, pages 702–721. Springer, 2020.

[26] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842, 2015.

[27] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[28] Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*, 2020.

[29] Fu-Ming Guo, Sijia Liu, Finlay S Mungall, Xue Lin, and Yanzhi Wang. Reweighted proximal pruning for large-scale language representation. *arXiv preprint arXiv:1909.12486*, 2019.

[30] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016.

[31] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[32] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[35] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

[36] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[37] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[38] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[39] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[40] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[41] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018.

[42] Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach. Structured sparse principal component analysis. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 366–373, 2010.

[43] Jianxin Wu Jian-Hao Luo and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, pages 5058–5066, 2017.

[44] Ian Jolliffe. *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[45] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[46] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *Acm transactions on graphics (tog)*, 22(3):277–286, 2003.

[47] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[48] Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In *European Conference on Computer Vision*, pages 639–654. Springer, 2020.

[49] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[50] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8018–8027, 2020.

[51] Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Exploiting kernel sparsity and entropy for interpretable cnn compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2019.

[52] Shaohui Lin, Rongrong Ji, Yuchao Li, Cheng Deng, and Xuelong Li. Toward compact convnets via structure-sparsity regularized filter pruning. *IEEE transactions on neural networks and learning systems*, 31(2):574–588, 2019.

[53] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in neural information processing systems*, pages 3288–3298, 2017.

[54] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[55] Fanxu Meng, Hao Cheng, Ke Li, Huixiang Luo, Xiaowei Guo, Guangming Lu, and Xing Sun. Pruning filter in filter. *arXiv preprint arXiv:2009.14410*, 2020.

[56] Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P Vetrov. Structured bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems*, pages 6775–6784, 2017.

[57] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017.

[58] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[59] Julie Nutini, Mark Schmidt, and Warren Hare. "active-set complexity" of proximal gradient: How long does it take to find the sparsity pattern? *Optimization Letters*, 13(4):645–655, 2019.

[60] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[61] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[62] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[63] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

[64] Victor Sanh, Thomas Wolf, and Alexander M Rush. Movement pruning: Adaptive sparsity by fine-tuning. *arXiv preprint arXiv:2005.07683*, 2020.

[65] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. Hydra: Pruning adversarially robust neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 7, 2020.

[66] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[67] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *arXiv preprint arXiv:2010.10732*, 2020.

[68] Frederick Tung and Greg Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[69] Mart van Baalen, Christos Louizos, Markus Nagel, Rana Ali Amjad, Ying Wang, Tijmen Blankevoort, and Max Welling. Bayesian bits: Unifying quantization and pruning. *arXiv preprint arXiv:2005.07093*, 2020.

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[71] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018.

[72] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *arXiv preprint arXiv:1608.03665*, 2016.

[73] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[74] Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010.

[75] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[76] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[77] Haichuan Yang, Shupeng Gui, Yuhao Zhu, and Ji Liu. Automatic neural network compression by sparsity-quantization joint learning: A constrained optimization-based approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2178–2188, 2020.

[78] Haiqin Yang, Zenglin Xu, Irwin King, and Michael R Lyu. Online learning for group lasso. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1191–1198, 2010.

[79] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *Advances in neural information processing systems*, pages 9793–9803, 2018.

[80] Cun-Hui Zhang et al. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942, 2010.

[81] Junyu Zhang and Lin Xiao. Multi-level composite stochastic optimization via nested variance reduction. *arXiv preprint arXiv:1908.11468*, 2019.

[82] Pengchuan Zhang, Hunter Lang, Qiang Liu, and Lin Xiao. Statistical adaptive stochastic gradient methods. *arXiv preprint arXiv:2002.10597*, 2020.

[83] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[84] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018.

[85] Kai Zhong, Zhao Song, Prateek Jain, Peter L Bartlett, and Inderjit S Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *International Conference on Machine Learning*, 2017.

[86] Yuefu Zhou, Ya Zhang, Yanfeng Wang, and Qi Tian. Accelerate cnn via recursive bayesian pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3306–3315, 2019.

[87] Tao Zhuang, Zhixuan Zhang, Yuheng Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning using polarization regularizer. *Advances in Neural Information Processing Systems*, 33, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] In the abstract and introduction, we highlighted the scope of this work and summarized the main contributions: (i) OTO, (ii) ZIG, (iii) HSPG, and (iv) CNN and Bert experiments.
   (b) Did you describe the limitations of your work? [Yes] See Section 5. OTO does not proceed quantization jointly yet.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This work focuses on a general neural network training and pruning framework. We did not expect any negative societal impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] As the supplementary materials to the proposed HSPG, we described and discussed the main theorems with the full set of assumptions in Appendix B.
   (b) Did you include complete proofs of all theoretical results? [Yes] In Appendix B, we included the complete proofs to the main theorems.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provided our source code, detailed data information, and instructions in supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We included the training details in Appendix A.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In Appendix A.3, we reported the error bars for all the main results shown in Section 4.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4.
   (b) Did you mention the license of the assets? [Yes] See Section 4.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] For the extensible experiments shown in Appendix B, we used several datasets from LIBSVM and the Fashion-MNIST with proper citation and description.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We have mentioned that all used datasets are publicly available to academic usage in Section 4.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We described the used datasets in Section 4 and concluded that they do not include personally identifiable information and offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]