DiSparse: Disentangled Sparsification for Multitask Model Compression

Xinglong Sun[1], Ali Hassani[1], Zhangyang Wang[2], Gao Huang[3], Humphrey Shi[1,4]
[1]SHI Lab @ UIUC & U of Oregon, [2]UT Austin,[3]Tsinghua University, [4]Picsart AI Research (PAIR)

Abstract

*Despite the popularity of Model Compression and Multitask Learning, how to effectively compress a multitask model has been less thoroughly analyzed due to the challenging entanglement of tasks in the parameter space. In this paper, we propose **DiSparse**, a simple, effective, and first-of-its-kind multitask pruning and sparse training scheme. We consider each task independently by disentangling the importance measurement and take the unanimous decisions among all tasks when performing parameter pruning and selection. Our experimental results demonstrate superior performance on various configurations and settings compared to popular sparse training and pruning methods. Besides the effectiveness in compression, DiSparse also provides a powerful tool to the multitask learning community. Surprisingly, we even observed better performance than some dedicated multitask learning methods in several cases despite the high model sparsity enforced by DiSparse. We analyzed the pruning masks generated with DiSparse and observed strikingly similar sparse network architecture identified by each task even before the training starts. We also observe the existence of a "watershed" layer where the task relatedness sharply drops, implying no benefits in continued parameters sharing. Our code and models will be available at: https://github.com/SHI-Labs/DiSparse-Multitask-Model-Compression.*

1. Introduction

Convolutional Neural Networks (CNNs) [29] are considered the go-to architecture for computer vision, ever since the inception of AlexNet [28], especially in fundamental vision tasks such as image classification [8], object detection [22, 35] and segmentation [3, 38]. As more complex and difficult vision tasks are explored, substantial efforts are devoted to scaling deep convolutional networks to enormous sizes. Many models exist with parameters as many as billions, which significantly challenges those targeting edge device applications. Therefore, effectively compressing deep convolutional networks for efficient storage and computation has been a very active research area, and various approaches have been proposed and developed [6, 9] over the years.

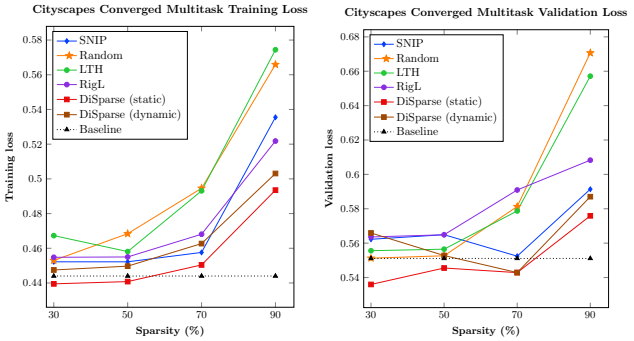

Figure 1. Converged training and validation multitask loss on Cityscapes. Our method DiSparse obtains the best training and validation behavior in both static and dynamic sparse training paradigms comparing to other methods. DiSparse even beats the unsparsified baseline at several sparsity levels.

Generally, neural network compression techniques can be categorized [9] into pruning [20, 30, 32], quantization [4, 37, 56], low-rank factorization [10, 33, 58], and knowledge distillation [25, 26, 36]. Network pruning, as a popular subfield of model compression, aims to discard certain parameters in the model, while retaining performance as much as possible. Pruning methods usually try to assign the best saliency score (also referred to as importance score) to each parameter and perform selection and pruning based on these importance measurements. Despite the diversity in the pruning schemes proposed in recent years, chasing sparsity in the network, either in a structured or unstructured manner, has been one of the central themes since the very beginning [21, 30]. Parameter efficiency of sparse neural networks has been demonstrated [19, 52], and multiple works [15, 48] have shown inference time speedups are possible using sparsity for convolutional neural networks.

Notwithstanding the increasing attention on model pruning and sparse training, effectively sparsifying a multitask network remains unexplored in spite of its importance. Multitask Learning (MTL) focuses on simultaneously solving multiple related tasks using a single model, which can significantly reduce the training and inference time and im-

prove the generalization performance through learning a shared representation across tasks [2]. It has a wide application in many problems like autonomous driving and indoor navigation robot where multiple tasks like semantic segmentation and depth estimation need to be performed simultaneously. A compact and efficient multitask network makes real-time performance possible on edge devices where computational resources are limited. Multitask network naturally brings storage and speed advantages over its single-task counterparts by a commonly shared backbone adopted in many popular MTL works [12, 27, 49]. However, how to further compress and sparsify such networks hasn't been carefully analyzed. Compression on multitask networks with comparable performance as on single-task ones is very challenging because the shared space contains heavily entangled and intertwined features, causing the traditional pruning and sparse training algorithm to fail. A fraction of parameters in the shared space, though not important for one task, could be crucial for the performance of another. A few MTL works have explored the problem of entangled features and showed disentangling representation into shared and task-private spaces will improve the model performance [34, 55].

We propose the first-of-its-kind pruning scheme that enforces sparsity in multitask networks by taking the entangled nature of their features into consideration. We argue that the key to properly compressing a multitask model is correctly identifying saliency scores for each task in the shared space, therefore **Spars**ifying in a **Di**sentangled manner (**DiSparse**). We take unanimous selection decisions among all tasks, which means that a parameter is removed only if it's shown to be not critical for any task. This prevents extreme degradation in performance for certain tasks due to sparsification, leading to a more balanced network.

We conduct extensive experiments to validate the efficiency of our proposed scheme. We demonstrate compression performance on models of different structures with datasets of various sizes [7, 46, 57]. We show results on pre-trained network, static sparse network at initialization, and dynamically growing sparse network at initialization. For each paradigm, we offered a slightly altered variant with the same core idea. We compared with popular pruning and sparse training methods and found that our proposed method is superior in both the training and validation phase, attaining lower training loss in a shorter period of time and better evaluation metrics on the validation set across different sparsity levels. Even more surprisingly, we observed better performance than some dedicated multitask learning approaches [1, 41, 50] in several cases in spite of the high sparsity enforced in our model, showing the effectiveness of DiSparse in multitask training. Besides the demonstration of superior model performance, we provide interesting observations in our experiments with DiSparse. We compute the Intersection over Union (IoU) of the binary pruning masks generated with DiSparse for each task to indicate task relatedness or similarity. Surprisingly, we observed strikingly similar sparse network architecture identified by each task even before the training starts. This offers a glimpse of the transferable subnetwork architecture across domains. Moreover, we observe the existence of a "watershed" layer where the task relatedness sharply drops, implying no benefits in continued parameters sharing. Exploitation of such property with DiSparse could save tremendous labor and computation cost by obtaining a better multitask architecture pre-training. These observations show that DiSparse does not only provide the compression community with the first-of-its-kind multitask sparsification scheme but also a powerful tool to the multitask learning community.

Our contributions can be summarized as follows:

- Proposing a simple, effective, and first-of-its-kind pruning and sparse training scheme for multitask network by disentangling the importance measurements among tasks, leading to a more balanced network.
- Performing an extensive empirical study on multiple vision tasks and datasets, which demonstrates the superiority of DiSparse compared to popular pruning and sparse training algorithms and even several dedicated multitask learning methods.
- Studying and discussing task relatedness and multitask model architecture design with DiSparse, which provides a valuable tool to the multitask learning community from a compression perspective.

2. Related Works

2.1. Pruning and Sparse Training

Network pruning is effective in reducing inference cost and storage. Pruning can be roughly categorized into two categories by architecture: unstructured and structured pruning. Unstructured pruning methods [20, 30] drop less significant weights, regardless of where they occur. On the other hand, structured pruning methods [32, 36], operate under structural constraints, for example removing convolutional filters or attention heads [40], thus enjoy immediate performance improvement without specialized hardware or library support. Pruning methods compute importance scores with different criterion to perform parameter selection. Most commonly used score criterion include: 1. Magnitude-based [20, 32], 2. Gradient-based [43, 44], 3. Hessian-based [21, 30], 4. Learning-based [11, 36]. Most pruning methods are only applied to pre-trained models.

Sparse training techniques, on the other hand, train sparse networks from scratch, and have also gained significant attention from the research community. The Lottery Ticket Hypothesis (LTH) [17] hypothesized that if we

can find a sparse neural network with iterative pruning, then we can train that sparse network from scratch to the same level of accuracy, by starting from the same initial conditions. Single-Shot Network Pruning (SNIP) [31] attempts to find an initial mask in a data-driven approach with one-shot pruning and uses this initial mask to guide parameters selection. As these methods still maintain a static network architecture throughout the training process, we refer to this group of methods as Static Sparse Training.

Another direction of sparse training is making network connections dynamic and allowing pruned weights to grow back. This allows for adaptive identification of high-quality sparse subnetworks. Sparse Evolutionary Training (SET) [42] prunes weights according to the standard magnitude criterion then adds weights back at random. Evci *et al*. [16] proposed Rigging the Lottery (RigL), an idea similar to SET but grows the weights back based on their gradients. We refer to this group of methods as Dynamic Sparse Training due to the model's ability to dynamically grow back pruned weights.

2.2. Multitask Network Compression

Research in multitask network compression has recently gained attention. Several methods [5, 23, 24] start with single-task networks and gradually merge them into a unified one, using feature sharing and similarity maximization. However, these schemes are inapplicable to pre-designed multitask models. This motivated our work, in which we propose a method that provides a pruning and sparse training scheme targeting a unified multitask network with shared parameters between tasks. To the best of our knowledge, our proposed method is the first to do so.

3. Methodology

3.1. Notations

Given a dataset $\mathcal{D}$ with individual samples x_i, targets y_i, and a desired sparsity level $\mathcal{S} \in (0, 1)$ (*i.e*. the ratio of zero weights), pruning or sparse training a neural network can be written as the following optimization problem:

$$\min_{\Theta} L(\Theta; \mathcal{D}) = \min_{\Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\Theta; x_i), y_i) \qquad (1)$$
$$\text{s.t.} \quad \Theta \in \mathbb{R}^m, \quad \|\Theta\|_0 \leq (1 - \mathcal{S}) \cdot m$$

Here, $\ell(.)$ is the standard loss function, Θ is the set of parameters of the neural network, m is the total number of parameters in the model, and $\|.\|_0$ represents the L_0 norm. For the ease of later notations, we reformulate the pruning and sparse training problem to find an optimal binary mask and modify Equation (1) as follows:

$$\min_{\mathcal{B}} L(\Theta, \mathcal{B}; \mathcal{D}) = \min_{\mathcal{B}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\Theta \odot \mathcal{B}; x_i), y_i) \qquad (2)$$
$$\text{s.t.} \quad \Theta \in \mathbb{R}^m, \mathcal{B} \in \mathbb{R}^m, \mathcal{B} \in \{0, 1\}^m, \|\mathcal{B}\|_0 \leq (1 - \mathcal{S}) \cdot m$$

where $\mathcal{B}$ represents a binary mask over the parameters, indicating which are kept and which are pruned. Given a set of $\mathcal{K}$ tasks $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_\mathcal{K}\}$, we represent the parameters used only by the k^{th} task as $\Theta^k \in \mathbb{R}^{m_k}$ and the common parameters used by all of the tasks as $\Theta^c \in \mathbb{R}^{m_c}$. We also denote $\Theta^k \cup \Theta^c$ as Θ^{kc}, and $\mathcal{B}^{kc}$ as its corresponding mask. Therefore masked parameters in the sparse network for task $\mathcal{T}_k$ can be expressed as $\Theta^{kc} \odot \mathcal{B}^{kc}$. Furthermore, the target for k^{th} task is denoted with y_i^k for the i^{th} data sample, as well as the loss function the same task with ℓ^k. As a result, total loss for all tasks is expressed as follows:

$$\ell(f(\Theta; x_i), y_i) = \sum_{k}^{\mathcal{K}} \lambda^k \ell^k(f(\Theta^{kc} \odot \mathcal{B}^{kc}; x_i), y_i^k) \qquad (3)$$

where λ^k represents the weighting scalar for each task. Equation (2) can therefore be re-written as:

$$\min_{\mathcal{B}} L(\Theta, \mathcal{B}; \mathcal{D}) = \min_{\mathcal{B}} \frac{1}{n} \sum_{i=1}^{n} \sum_{k}^{\mathcal{K}} \lambda^k \ell^k(f(\Theta^{kc} \odot \mathcal{B}^{kc}; x_i), y_i^k)$$
$$= \min_{\mathcal{B}} \sum_{k}^{\mathcal{K}} L^k(\Theta^{kc} \odot \mathcal{B}^{kc}; \mathcal{D}) \qquad (4)$$
$$\text{s.t.} \quad \Theta \in \mathbb{R}^m, \mathcal{B} \in \mathbb{R}^m, \mathcal{B} \in \{0, 1\}^m, \|\mathcal{B}\|_0 \leq (1 - \mathcal{S}) \cdot m$$

3.2. Our Method

When solving for task-specific binary masks, $\mathcal{B}^k$, traditional pruning or sparse training methodologies work as proposed. However, while solving for $\mathcal{B}^c$, the binary mask for the large number of commonly shared parameters, we can't simply apply typical methods which directly utilize $L(\Theta, \mathcal{B}; \mathcal{D})$ as guidance, because the shared parameters are entangled with multiple tasks. Therefore, we propose a scheme for solving the shared mask, $\mathcal{B}^c$. For a given task $\mathcal{T}_k$, we compute the binary mask $\mathcal{B}^{kc}$ solely based on the task itself. This means saliency scores are computed solely based on $L^k(\Theta^{kc}, \mathcal{B}^{kc}; \mathcal{D})$. This mask will capture the preferred sub-network structure provided that we solve for task $\mathcal{T}_k$ independently. We denote the mask corresponding to the shared parameters in $\mathcal{B}^{kc}$ as $\mathcal{C}(\mathcal{B}^{kc}) \in \mathbb{R}^{m_c}$ and the task-private parameters in $\mathcal{B}^{kc}$ as $\mathcal{P}(\mathcal{B}^{kc}) \in \mathbb{R}^{m_k}$. We can then make the direct assignment $\mathcal{B}^k = \mathcal{P}(\mathcal{B}^{kc})$. To solve for the mask for the shared parameters, B^c, we feed all of the masks into an "arbiter" function, $\mathcal{A}$, and compute the final mask $\mathcal{B}^c$ for the shared parameters shown as follows:

$$\mathcal{B}^c = \mathcal{A}(\mathcal{C}(\mathcal{B}^{1c}), \mathcal{C}(\mathcal{B}^{2c}), \ldots, \mathcal{C}(\mathcal{B}^{\mathcal{K}c})) \qquad (5)$$

Algorithm 1 Proposed Algorithm: DiSparse

Input: $\Theta^{kc} \in \mathbb{R}^{m_k+m_c}, L^k(.) \quad \forall k \in \{1, 2, \ldots, \mathcal{K}\}$
$\mathcal{A}, \mathcal{S}, \mathcal{D}, saliency(.)$
Output: $\mathcal{B}$

$\mathcal{B}cs \leftarrow [\ ]$
for $k \leftarrow 1$ to $\mathcal{K}$ **do**
  $\vec{sk} \leftarrow saliency(\Theta^{kc}, L^k(.), \mathcal{D})$
  $\gamma \leftarrow (1 - S) \cdot (m_k + m_c)$
  $\tilde{sk}\gamma \leftarrow Top\gamma(\vec{sk})$
  $\mathcal{B}kc \leftarrow \vec{sk} > \tilde{sk}\gamma$ `//Element-wise Comparison`
  $\mathcal{B}k \leftarrow \mathcal{P}(\mathcal{B}kc)$ `//Task-specific Mask`
  $\mathcal{B}cs.append(\mathcal{C}(\mathcal{B}kc))$ `//Common Mask`
end for
$\mathcal{B}c \leftarrow \mathcal{A}(\mathcal{B}cs)$
$\mathcal{B} \leftarrow \mathcal{B}c \cup (\bigcup_{k=1}^{\mathcal{K}} \mathcal{B}k)$
return $\mathcal{B}$

We use three variants of saliency score criterion based on our proposed scheme for three different popular paradigms to enforce sparsity in the model: 1). Static Sparse Training 2). Dynamic Sparse Training 3). Pruning on pre-trained models. In the following subsections, we will outline how $\mathcal{B}^{kc}$ is computed for each paradigm.

3.2.1 Static Sparse Training

We borrowed the effective data-driven sensitivity measurement from SNIP [31] and made modifications based on our scheme. We represent the saliency for task $\mathcal{T}_k$ of the j^{th} weight in the parameter space Θ^{kc}, Θ^{kc}_j, as s^{kc}_j, which is computed as follows:

$$g^{kc}_j(\Theta^{kc}; \mathcal{D}) = \frac{\partial L^k(\Theta^{kc} \odot \mathcal{B}^{kc}; \mathcal{D})}{\partial \mathcal{B}^{kc}_j} \tag{6}$$

$$s^{kc}_j = \frac{|g^{kc}_j(\Theta^{kc}; \mathcal{D})|}{\sum_z^{m_c} |g^{kc}_z(\Theta^{kc}; \mathcal{D})|} \tag{7}$$

We can then use the saliency measurements to compute the mask $\mathcal{B}^{kc}$ for task $\mathcal{T}_k$.

$$\gamma = (1 - S) \cdot (m_k + m_c) \tag{8}$$

$$\mathcal{B}^{kc}_j = \mathbb{I}[s^{kc}_j - s^{\tilde{kc}}_{\gamma} \geq 0], \quad \forall j \in \{1 \ldots m_k + m_c\} \tag{9}$$

Here, $s^{\tilde{kc}}_{\gamma}$ is the γ^{th} largest element in the vector $s^{\vec{kc}}$.
We can thus use $\mathcal{B}^{kc}$ as discussed in 3.2 to compute the final mask for the entire model.

3.2.2 Dynamic Sparse Training

We follow RigL [16] in terms of the pruning and growing criterion and made modifications based on our scheme. For pruning, we performed the standard magnitude pruning. For growing, as in 3.2.1, we compute the saliency for task $\mathcal{T}_k$ of the j^{th} weight in the parameter space, s^{kc}_j, as follows:

$$g^{kc}_j(\Theta^{kc}; \mathcal{D}) = \frac{\partial L^k(\Theta^{kc}; \mathcal{D})}{\partial \Theta^{kc}_j} \tag{10}$$

$$s^{kc}_j = |g^{kc}_j(\Theta^{kc}; \mathcal{D})| \tag{11}$$

We can then use the saliency measurements to compute the growing mask $\mathcal{B}^{kc}$ for task $\mathcal{T}_k$.

$$\gamma = f_{decay}(t; \alpha, T_{end}) \cdot (1 - S) \cdot (m_k + m_c) \tag{12}$$

$$\mathcal{B}^{kc}_j = \mathbb{I}[s^{kc}_j - s^{\tilde{kc}}_{\gamma} \geq 0], \quad \forall j \text{ s.t. } \Theta^{kc}_j \notin \Theta^{kc} \setminus \mathbb{I}_{active} \tag{13}$$

Here, $f_{decay}(.)$ is a decaying function as in RigL [16] to update the ratio of parameters to prune and grow at each update iteration, and $\Theta^{kc} \setminus \mathbb{I}_{active}$ is the set of active connections remaining after pruning.
We can thus use $\mathcal{B}^{kc}$ as discussed in 3.2 to compute the final mask for the entire model.

3.2.3 Pruning on Pre-trained Models

We borrowed the pruning criterion from GF [39] and made modifications based on our scheme. As in 3.2.1, we compute the saliency for task $\mathcal{T}_k$ of the j^{th} weight in the parameter space, s^{kc}_j, as follows:

$$g^{kc}_j(\Theta^{kc}; \mathcal{D}) = \frac{\partial L^k(\Theta^{kc}; \mathcal{D})}{\partial \Theta^{kc}_j} \tag{14}$$

$$s^{kc}_j = |g^{kc}_j(\Theta^{kc}; \mathcal{D})| \cdot |\Theta^{kc}_j|^2 \tag{15}$$

Therefore, we can use the saliency measurements to compute the pruning mask $\mathcal{B}^{kc}$ for task $\mathcal{T}_k$.

$$\gamma = (1 - S) \cdot (m_k + m_c) \tag{16}$$

$$\mathcal{B}^{kc}_j = \mathbb{I}[s^{kc}_j - s^{\tilde{kc}}_{\gamma} \geq 0], \quad \forall j \in \{1 \ldots m_k + m_c\} \tag{17}$$

We can thus use $\mathcal{B}^{kc}$ as discussed in 3.2 to compute the final mask for the entire model.

3.2.4 Arbiter Function

As discussed in 3.2, we used an "arbiter" function, $\mathcal{A}$, to solve the mask for the common parameters, $\mathcal{B}^c$. We provided two choices here:
Element-wise Logical OR. We performed a logical OR operation over the mask computed by each task. In this case, a parameter will be pruned only if it's considered not important for each task.

$$\begin{aligned} \mathcal{B}^c &= \mathcal{A}(\mathcal{C}(\mathcal{B}^{1c}), \mathcal{C}(\mathcal{B}^{2c}) \ldots \mathcal{C}(\mathcal{B}^{\mathcal{K}c})) \\ &= \mathcal{C}(\mathcal{B}^{1c}) | \mathcal{C}(\mathcal{B}^{2c}) | \ldots | \mathcal{C}(\mathcal{B}^{\mathcal{K}c}) \end{aligned} \tag{18}$$

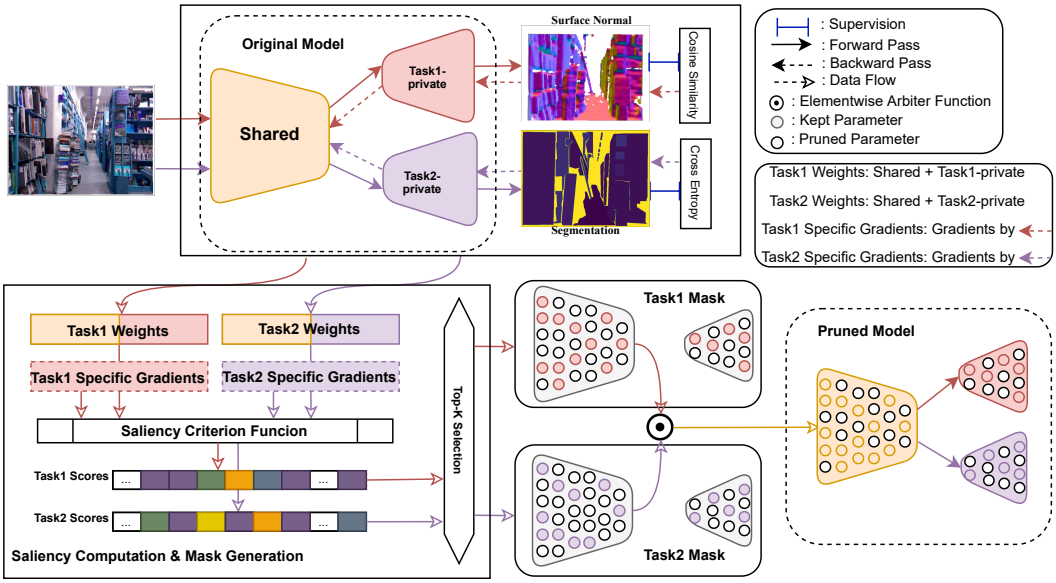

Figure 2. An overview of DiSparse . For task $\mathcal{T}_k$, we feed weights Θ^{kc} and their gradients w.r.t the loss $L^k(.)$ into a saliency scoring function to get their importance scores. Later we generate an optimal binary mask $\mathcal{B}^{kc}$ for the model assuming that we're only training the network independently for task $\mathcal{T}_k$. We directly assign $\mathcal{P}(\mathcal{B}^{kc})$, the task-private part, to $\mathcal{B}^k$ used as the pruning or growing mask for the task-private parameters. For $\mathcal{C}(\mathcal{B}^{kc})$, the shared part, we feed all of $\{\mathcal{C}(\mathcal{B}^{kc}), \forall k \in \{1, \ldots, \mathcal{T}\}\}$ to an element-wise arbiter function $\mathcal{A}$ and take its output as $\mathcal{B}^c$, the pruning or growing mask for the shared parameters.

Majority Vote. For three or more tasks ($\mathcal{K} \geq 3$), element-wise majority vote can be applied for more effective compression. A parameter will be pruned if most of the tasks agree to remove this particular connection.

$$\begin{aligned} \mathcal{B}^c &= \mathcal{A}(\mathcal{C}(\mathcal{B}^{1c}), \mathcal{C}(\mathcal{B}^{2c}) \ldots \mathcal{C}(\mathcal{B}^{\mathcal{K}c})) \\ &= MAJ(\mathcal{C}(\mathcal{B}^{1c}), \mathcal{C}(\mathcal{B}^{2c}) \ldots \mathcal{C}(\mathcal{B}^{\mathcal{K}c})) \end{aligned} \tag{19}$$

4. Empirical Evaluation

We conduct extensive experiments to show that our proposed method outperforms many strong baselines and is very effective in sparsifying a multitask model.

4.1. Datasets, Tasks, and Model

We evaluate our method on three popular multitask datasets: NYU-v2 [46], Cityscapes [7], and Tiny-Taskonomy [57]. We perform joint Semantic Segmentation and Surface Normal Prediction [18,41,54] for NYU-v2, and joint Semantic Segmentation and Depth Prediction [54] for Cityscapes. For Tiny-Taskonomy, we perform joint training on 5 tasks: Semantic Segmentation, Surface Normal Prediction, Depth Prediction, Keypoint Detection, and Edge Detection [53, 54].

We performed sparsification on the widely-used DeepLab-ResNet [3] with atrous convolutions as the backbone and the ASPP [3] architecture for task-specific heads. As discussed in the above sections, we use the popular multitask architecture, in which all tasks share the backbone but have separate task-specific heads.

4.2. Evaluation Metrics

We demonstrated sparsity level as the most direct evaluation metric for the pruned or the sparse-trained model. The higher the sparsity level of a model, the more zeros it contains thus enjoying more acceleration and memory benefits. In terms of the performance evaluation of tasks, for Semantic Segmentation, we show mean Intersection over Union (mIoU) and Pixel Accuracy (Pixel Acc). For Surface Normal Prediction, we use mean and median angle error as the main evaluation metrics. On NYU-v2, we also demonstrate the percentage of pixels whose prediction is within the angles of $11.25°$, $22.5°$, and $30°$ to the ground truth [13]. For Depth Prediction, we compute absolute and relative errors as the main evaluation metrics. On Cityscapes, we also show the relative difference between the prediction and ground truth via the percentage of $\delta = max(\frac{y_{pred}}{y_{gt}}, \frac{y_{gt}}{y_{pred}})$ within threshold 1.25, 1.25^2, and 1.25^3 [14]. In both keypoints and edge detection tasks, we choose to show the mean absolute error to the provided ground-truth map as the main evaluation metric. Since for multitask learning, it's hard to qualify the performance of a model or algorithm with a single metric, we also demonstrate the converged training and testing multitask loss to show how our proposed scheme helps the model to learn in the training

Model	T1: Semantic Seg.		T2: SN Prediction					Sparsity	Pre-trained
	mIoU↑	PixelAcc↑	Mean Err↓	Median Err↓	11.25↑	22.5↑	30↑	(%)↑	
DeepLab [3](baseline)	27.69	58.77	16.55	14.17	39.62	73.54	86.33	0	N/A
LTH [17]	24.63	56.25	17.01	14.27	39.49	71.95	84.31	90.00	✓
SNIP [31]	23.83	56.90	16.58	14.05	39.82	73.73	85.77	90.00	✗
Random	25.56	25.18	18.86	16.22	27.81	69.55	85.45	90.00	✗
DiSparse (*Ours*)	**26.48**	**57.77**	**16.44**	**13.69**	**41.24**	**74.07**	**85.85**	90.00	✗
RigL [16]	24.91	56.74	17.27	14.63	38.52	70.75	83.71	90.00	✗
DiSparse (*Ours*)	**28.16**	**59.18**	**16.54**	**13.47**	**42.25**	**73.14**	**84.73**	90.00	✗
IMP [20]	29.19	59.81	16.58	13.32	43.32	72.31	84.03	90.00	✓
Random	25.56	25.18	18.69	16.04	23.37	**73.08**	**86.78**	90.00	✓
DiSparse (*Ours*)	**29.45**	**59.95**	**16.56**	**13.30**	**43.33**	72.49	84.18	90.00	✓

Table 1. DiSparse semantic segmentation and surface normal prediction results on NYU-v2 [46] compared to static sparse training, dynamic sparse training, and pre-trained model pruning methods.

phase.

4.3. Experimental Settings

We used PyTorch for all of our experiments, and two RTX 2080 Ti GPUs. We used Adam optimizer and a batch size of 16 for all experiments. We trained NYU-v2 for $20K$ iterations and used an initial learning rate of $1e{-}3$, decaying by 0.5 every 4000 iterations. We used the same optimization settings for Cityscapes, except for the initial learning rate, which was set to $1e-4$. Tiny-Taskonomy was trained for 100K iterations with an initial learning rate of $1e-4$ decaying by 0.3 every 12K iterations. We used cross-entropy loss for Semantic Segmentation, negative cosine similarity between the normalized prediction and ground-truth for Surface Normal Prediction, and L1 loss for the rest of the tasks. To avoid bias and diversity in different pre-trained models, we train all of the models from scratch for a fair comparison among different methods.

4.4. Baselines and Ours

We evaluated several methods: LTH [17], RigL [16], SNIP [31], and IMP [20]. We used SNIP's official implementation in Tensorflow, and implemented the rest in PyTorch. For LTH [17], we used the fully-trained model to get the sub-network structure and rewind the model to the initial weights to start the sparse training. For SNIP [31], we use the gradients on 50 random data batches drawn from the training datasets to compute the sparse mask. For RigL [16], we followed the original paper and utilized a cosine decay to gradually decrease the pruning and growing ratio at each iteration. We stopped the update at the 75%-th iteration. Also, for the initial sparsity distribution over layers, we adopted *Erdős-Rényi-Kernel (ERK)* introduced in [16, 42] as it was shown to attain the best performance. For our method, we used the same configurations as SNIP in static sparse training experiments and RigL in dynamic sparse training ones. In terms of the optimization, for the sparse training methods, we used the same settings as the original models described in 4.3. For the pruning on pre-trained model experiments, we retrained the model for 1000 iterations with learning rate set as $1e-5$ after pruning for NYU-v2 and Cityscapes. For Tiny-Taskonomy, we re-trained for 4000 iterations with learning rate set as $1e-6$ decaying by 0.3 every 1500 iterations. Also, we select the Element-wise OR as our arbiter function for results reported in the tables but also explore the Majority Vote choice in the ablation studies.

4.5. Quantitative Results

We demonstrated results in three different learning scenarios and datasets in Table 1, 2, and 3. We performed all of the experiments on four different sparsity levels $(30\%, 50\%, 70\%, 90\%)$ for all of the methods. Due to the space constraint, we only report evaluation results at the most extreme sparsity level, 90%. However, we do show the converged training and validation loss for all of the sparse training methods including two of ours at all sparsity levels on Cityscapes in Figure 1. From the figure, the superiority of our proposed scheme is clearly observed in both the training and validation set. It attains much lower training loss across all sparsity levels than other methods and generalizes well on the validation set. For full evaluation results at all sparsity levels, please refer to the Appendix. In the tables, we reported all the evaluation metrics discussed in 4.2. We also indicate whether the method utilized any information from the pre-trained model. Though LTH [17] proposes a sparse training approach, it relies on the pre-trained model to extract the sub-network architecture. As shown in [59], the "winning lottery tickets" obtain non-random accuracies even before training has started. Thus, we also check the "pre-trained" mark for LTH in our tables. For all the learning scenarios, we also include random pruning or sparse training results.

On NYU-v2, as shown in Table 1, DiSparse outper-

Model	T1: Semantic Seg.			T2: Depth Prediction					Sparsity	Pre-trained
	mIoU ↑	PixelAcc ↑	Error ↓	Abs. Error ↓	Rel. Error ↓	$\delta 1.25$ ↑	$\delta 1.25^2$ ↑	$\delta 1.25^3$ ↑	(%) ↑	
DeepLab [3](baseline)	42.58	74.84	0.49	0.016	0.33	74.22	88.90	94.47	0	N/A
LTH [17]	36.19	73.97	0.51	0.017	**0.35**	72.46	87.29	93.59	90.00	✓
SNIP [31]	**38.47**	73.99	0.53	0.016	0.36	72.49	87.51	**93.61**	90.00	✗
Random	35.67	71.74	0.60	0.021	0.43	66.47	82.21	90.63	90.00	✗
DiSparse (*Ours*)	38.21	**74.31**	**0.49**	**0.015**	**0.35**	**72.72**	**87.60**	93.59	90.00	✗
RigL [16]	36.57	74.02	0.51	0.017	0.37	71.98	86.66	93.05	90.00	✗
DiSparse (*Ours*)	**38.55**	**74.28**	**0.49**	**0.016**	**0.36**	**72.11**	**87.01**	**93.41**	90.00	✗
IMP [20]	37.63	73.90	0.50	0.018	**0.37**	71.15	86.41	93.01	90.00	✓
Random	33.62	51.63	1.18	0.034	0.43	50.09	72.42	82.73	90.00	✓
DiSparse (*Ours*)	**40.71**	**74.64**	**0.49**	**0.016**	**0.37**	**72.06**	**86.67**	**93.11**	90.00	✓

Table 2. DiSparse semantic segmentation and depth prediction results on Cityscapes [7] compared to static sparse training, dynamic sparse training, and pre-trained model pruning methods.

forms all the other methods by a significant margin in all paradigms. In the static sparse training paradigm, DiSparse gets really close overall performance to the original unsparsified model and even better performance on three metrics (*i.e.* Median Angle Error, $11.25°$, and $22.5°$). As discussed in 3.2, we borrowed the saliency measurement from SNIP [31] for static sparse training. As seen in the table, DiSparse is much more effective compared to SNIP. In the dynamic training paradigm, DiSparse performs much better than RigL [16]. Surprisingly, our performance is even better than the original unsparsified model by a noticeable margin, demonstrating the efficacy of our proposed scheme. In fact, the performance of DiSparse surpasses several dedicated multitask learning methods [1, 41, 50] in the same optimization settings but with zero sparsity. Comparison with such methods is included in the Appendix. On the paradigm where pruning is performed on pre-trained models, DiSparse is also better than both IMP [20] and random pruning by a noticeable margin.

On Cityscapes, as shown in Table 2, DiSparse is shown to be more superior than other methods consistently across all paradigms. On static sparse training, though SNIP obtains better mIoU and close Pixel Acc to DiSparse, it performs poorly on the depth prediction task. As another indicator of Semantic Segmentation task, we also include the mean absolute error indicated as "Error" in Table 2. As shown, DiSparse even obtains the same error as the baseline. On the dynamic sparse training and pruning paradigms, DiSparse outperforms other methods by a substantial performance margin across all metrics.

DiSparse is also superior in Tiny-Taskonomy, as shown in Table 3. Its performance is demonstrated to be more balanced across tasks. For example, although SNIP gets slightly better mIoU and Pixel Acc, it performs even much worse than the random sparse training on surface normal prediction. Whereas our method is demonstrated to have decent performance on all of the 5 tasks and does not have one task being drastically worse than others. On the dynamic sparse training and pruning paradigms, DiSparse is also shown to be much more effective than other methods.

The consistent superiority of DiSparse on all three datasets shows that it can indeed help multitask model achieve high sparsity with minimum performance drop.

4.6. Ablation Studies

We explored the effect of the arbiter function $\mathcal{A}$ and demonstrated the results in Table 4. As discussed above, results reported in Table 1, 2, and 3 come from methods with Element-wise OR as the arbiter function. Here, we tried using the Element-wise Majority Vote on Tiny-Taskonomy since it only makes sense when we have a relatively large number of tasks ($\mathcal{K} \geq 3$). Compared to results shown in Table 3, Majority Vote yields better results in the static sparse training paradigm. It gives better performance on Semantic Segmentation and Surface Normal Prediction. However, in the dynamic sparse training setup, it doesn't surpass the performance of the Element-wise OR option but is still better than RigL in overall performance.

4.7. Limitation

As shown in 3.2, the sparsity of our model will not be exactly as requested due to the operations taken by the arbiter function $\mathcal{A}$. For the dynamic sparse training experiments, since DiSparse grows more weights in the growing session, we simply adjust the pruning rate in the following pruning session to compensate for it. For example, suppose the expected sparsity level at iteration i is S_i and the expected pruning rate is P_i. Our method generates a model with sparsity $\hat{S}_i, \hat{S}_i \leq S_i$. We adjust the pruning rate to $\hat{P}_i = 1 - \frac{1-S_i}{1-\hat{S}_i} \cdot (1 - P_i)$ to keep the final sparsity as expected. For static sparse training, it's more complex and we set up an empirical approach. We start from the requested sparsity and iteratively increase it and run DiSparse until the final sparsity is within an acceptable margin. When the final sparsity is a little bit off, we always keep it higher than that of the baselines for a fair comparison.

Model	T1: Semantic Seg.		T2: SN Prediction		T3: Depth		T4: Keypt.	T5: Edge	Sparsity	Pre-trained
	mIoU ↑	PixelAcc ↑	Mean Err. ↓	Med. Err. ↓	Abs. Err ↓	Rel. Err ↓	Err ↓	Err ↓	(%) ↑	
DeepLab [3](baseline)	29.51	92.98	24.36	12.88	0.021	0.033	0.20	0.21	90.00	N/A
LTH [17]	24.36	91.62	26.07	15.78	0.023	0.036	**0.20**	0.22	90.00	✓
SNIP [31]	**26.03**	**91.89**	27.06	18.12	**0.022**	0.036	**0.20**	**0.21**	90.00	✗
Random	25.39	91.18	26.31	16.30	0.023	0.037	**0.20**	0.22	90.00	✗
DiSparse (*Ours*)	25.53	91.85	**25.42**	**14.80**	**0.022**	**0.035**	**0.20**	**0.21**	90.00	✗
RigL [16]	23.87	90.14	26.91	17.75	0.023	0.037	**0.19**	**0.20**	90.00	✗
DiSparse (*Ours*)	**25.21**	**91.49**	**25.28**	**14.63**	**0.022**	**0.034**	**0.19**	**0.20**	90.00	✗
IMP [20]	26.71	91.74	24.99	13.89	0.021	0.033	0.20	0.21	90.00	✓
Random	23.89	83.89	41.21	38.35	0.25	0.39	0.45	0.40	90.00	✓
DiSparse (*Ours*)	**29.12**	**92.58**	**24.70**	**13.42**	**0.021**	**0.033**	**0.19**	**0.20**	90.00	✓

Table 3. DiSparse semeantic segmentation, surface normal prediction, depth prediction, keypoint detection, and edge detection results on Tiny-Taskonomy [57] compared to static sparse training, dynamic sparse training, and pre-trained model pruning methods.

Model	T1: Semantic Seg.		T2: SN Prediction		T3: Depth		T4: Keypt.	T5: Edge	Sparsity	Pre-trained
	mIoU ↑	PixelAcc ↑	Mean Err. ↓	Med. Err. ↓	Abs. Err ↓	Rel. Err ↓	Err ↓	Err ↓	(%) ↑	
DiSparse (static)	26.03 (**0.50**)	92.03 (**0.18**)	25.28 (**-0.14**)	14.52 (**-0.32**)	0.022 (0.00)	0.035 (0.00)	0.20 (0.00)	0.22 (0.01)	90.00	✗
DiSparse (dynamic)	24.97 (-0.24)	91.77 (**0.28**)	25.89 (0.61)	15.88 (1.25)	0.023 (0.001)	0.036 (0.002)	0.20 (0.01)	0.21 (0.01)	90.00	✗

Table 4. Ablations on Tiny-Taskonomy [57]. We show the results with the majority vote. Values inside the parenthesis are changes from the corresponding metrics in Table 3. Positive changes are bold.

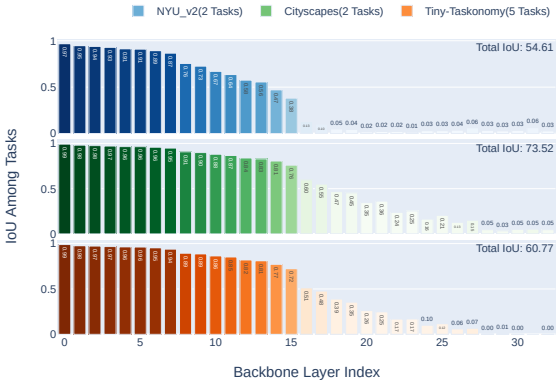

Figure 3. Layer-wise IoU evaluating preferred sparse subnetwork architecture among different tasks on three different datasets.

4.8. Discussion

LTH [17] inspired many to study the transferability of the "winning ticket", or sparse subnetwork architecture in general, across domains [45, 47, 51]. Morcos *et al.* [45] showed that winning ticket initializations generalized across a variety of natural image datasets, suggesting that different tasks seem to enjoy the same sparse subnetwork structure. Here, we made another exploratory step towards finding a transferable across domains sparse sub-network architecture. In the static sparse training setups, before the training even started, we analyzed the masks generated by different tasks regarding the commonly shared backbone with criterion discussed in 3.2.1. To evaluate the task similarity and relatedness, we compute Intersection over Union (IoU) between their computed masks, which indicate the agreement and divergence of the preferred sparse subnetwork architecture by different tasks. Similar to 3.2, we use $\mathcal{C}(\mathcal{B}^{kc})$ to denote the preferred mask on the shared backbone for task $\mathcal{T}_k$. IoU can therefore be computed as $IoU = \frac{|\bigcap_{k=1}^{\mathcal{K}} \mathcal{C}(\mathcal{B}^{kc})|}{|\bigcup_{k=1}^{\mathcal{K}} \mathcal{C}(\mathcal{B}^{kc})|}$. We present layer-wise IoU scores from the first 32 layers of the backbone on all three datasets in Figure 3. Surprisingly, we observed strikingly high IoU among tasks, even in the 5-task Tiny-Taskonomy dataset. This implies that even before training starts, different tasks tend to select the same architecture in the shared parameter space to facilitate training, suggesting potential for domain-independent sparse architecture exploration. Another interesting observation is that IoU drops sharply at certain layers. This phenomenon is observed across all three datasets, though this "watershed" layer is different (Layer 18 on NYU-v2, Layer 28 on Cityscapes, and Layer 26 on Taskonomy). This IoU analysis among tasks could be very helpful for pre-training multitask network design. For example, we could stop sharing parameters in the backbone and start branching out for different tasks at the layer where IoU sharply drops. We leave this for future discussions.

5. Conclusion

In this paper, we present DiSparse, a novel sparse training and pruning method targeting multitask models. We conduct extensive experiments which demonstrate its superiority compared to other related methods on three different datasets and learning scenarios. We also provide promising future research directions in transferrable lottery-ticket across domains and multitask architecture design with our proposed scheme.

References

[1] Chanho Ahn, Eunwoo Kim, and Songhwai Oh. Deep elastic networks with model selection for multi-task learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6529–6538, 2019. 2, 7

[2] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. 2

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2017. 1, 5, 6, 7, 8

[4] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning(ICML)*, pages 2285–2294. PMLR, 2015. 1

[5] Hanjing Cheng, Zidong Wang, Lifeng Ma, Xiaohui Liu, and Zhihui Wei. Multi-task pruning via filter index sharing: A many-objective optimization approach. *Cognitive Computation*, 13(4):1070–1084, 2021. 3

[6] 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. 1

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 2, 5, 7

[8] 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. 1

[9] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020. 1

[10] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Neural Information Processing Systems(NIPS)*, pages 1269–1277, 2014. 1

[11] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4510–4520, 2021. 2

[12] Nikita Dvornik, Konstantin Shmelkov, Julien Mairal, and Cordelia Schmid. Blitznet: A real-time deep network for scene understanding. In *Proceedings of the IEEE international conference on computer vision*, pages 4154–4162, 2017. 2

[13] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015. 5

[14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014. 5

[15] Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. Fast sparse convnets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14629–14638, 2020. 1

[16] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020. 3, 4, 6, 7, 8

[17] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018. 2, 6, 7, 8

[18] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3205–3214, 2019. 5

[19] 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. 1

[20] 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. 1, 2, 6, 7, 8

[21] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in Neural Information Processing Systems*, 5, 1992. 1, 2

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[23] Xiaoxi He, Dawei Gao, Zimu Zhou, Yongxin Tong, and Lothar Thiele. Pruning-aware merging for efficient multitask inference. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 585–595, 2021. 3

[24] Xiaoxi He, Zimu Zhou, and Lothar Thiele. Multi-task zipping via layer-wise neuron sharing. *arXiv preprint arXiv:1805.09791*, 2018. 3

[25] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *Neural Information Processing Systems (NIPS)*, 2014. 1

[26] Jianbo Jiao, Yunchao Wei, Zequn Jie, Honghui Shi, Rynson WH Lau, and Thomas S Huang. Geometry-aware distillation for indoor semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR*, pages 2869–2878, 2019. 1

[27] Brendan Jou and Shih-Fu Chang. Deep cross residual learning for multitask visual recognition. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 998–1007, 2016. 2

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 1

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[30] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990. 1, 2

[31] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2018. 3, 4, 6, 7, 8

[32] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations(ICLR)-Poster*, 2017. 1, 2

[33] Shaohui Lin, Rongrong Ji, Chao Chen, Dacheng Tao, and Jiebo Luo. Holistic cnn compression via low-rank decomposition with knowledge transfer. *IEEE transactions on pattern analysis and machine intelligence(TPAMI)*, 41(12):2889–2905, 2018. 1

[34] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*, 2017. 2

[35] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision (ECCV)*, pages 21–37. Springer, 2016. 1

[36] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 2604–2613, 2019. 1, 2

[37] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, pages 722–737, 2018. 1

[38] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[39] Ekdeep Singh Lubana and Robert Dick. A gradient flow framework for analyzing network pruning. In *International Conference on Learning Representations*, 2020. 4

[40] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*, 32:14014–14024, 2019. 2

[41] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016. 2, 5, 7

[42] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018. 3, 6

[43] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017. 2

[44] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019. 2

[45] Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *Advances in Neural Information Processing Systems*, 32:4932–4942, 2019. 8

[46] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2, 5, 6

[47] Michela Paganini and Jessica Zosa Forde. Bespoke vs. pr\^ et-\a-porter lottery tickets: Exploiting mask similarity for trainable sub-network finding. *arXiv preprint arXiv:2007.04091*, 2020. 8

[48] Jongsoo Park, Sheng R Li, Wei Wen, Hai Li, Yiran Chen, and Pradeep Dubey. Holistic sparsecnn: Forging the trident of accuracy, speed, and size. *arXiv preprint arXiv:1608.01409*, 1(2), 2016. 1

[49] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):121–135, 2017. 2

[50] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829, 2019. 2, 7

[51] Matthia Sabatelli, Mike Kestemont, and Geurts Pierre. On the transferability of winning tickets in non-natural image datasets. In *16th International Conference on Computer Vision Theory and Applications-VISAPP 2021*, 2021. 8

[52] Suraj Srinivas, Akshayvarun Subramanya, and R Venkatesh Babu. Training sparse neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 138–145, 2017. 1

[53] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020. 5

[54] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423*, 2019. 5

[55] Mihai Suteu and Yike Guo. Regularizing deep multi-task networks using orthogonal gradients. *arXiv preprint arXiv:1912.06844*, 2019. 2

[56] Haichao Yu, Haoxiang Li, Humphrey Shi, Thomas S Huang, and Gang Hua. Any-precision deep neural networks. In *AAAI*, 2021. 1

[57] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 2, 5, 8

[58] Xiangyu Zhang, Jianhua Zou, Xiang Ming, Kaiming He, and Jian Sun. Efficient and accurate approximations of nonlinear convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and pattern Recognition(ICCV)*, pages 1984–1992, 2015. 1

[59] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: zeros, signs, and the supermask. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 3597–3607, 2019. 6