Dynamic Slimmable Network

Changlin Li[1] Guangrun Wang[2] Bing Wang[3] Xiaodan Liang[4] Zhihui Li[5] Xiaojun Chang[1*]

[1] GORSE Lab, Dept. of DSAI, Monash University [2] Univeristy of Oxford [3] Alibaba Group

[4] Sun Yat-Sen University [5] Shandong Artificial Intelligence, Qilu University of Technology

changlin.li@monash.edu, wanggrun@gmail.com, fengquan.wb@alibaba-inc.com,

xdliang328@gmail.com, zhihuilics@gmail.com, xiaojun.chang@monash.edu

Abstract

Current dynamic networks and dynamic pruning methods have shown their promising capability in reducing theoretical computation complexity. However, dynamic sparse patterns on convolutional filters fail to achieve actual acceleration in real-world implementation, due to the extra burden of indexing, weight-copying, or zero-masking. Here, we explore a dynamic network slimming regime, named Dynamic Slimmable Network (DS-Net), which aims to achieve good hardware-efficiency via dynamically adjusting filter numbers of networks at test time with respect to different inputs, while keeping filters stored statically and contiguously in hardware to prevent the extra burden. Our DS-Net is empowered with the ability of dynamic inference by the proposed double-headed dynamic gate that comprises an attention head and a slimming head to predictively adjust network width with negligible extra computation cost. To ensure generality of each candidate architecture and the fairness of gate, we propose a disentangled two-stage training scheme inspired by one-shot NAS. In the first stage, a novel training technique for weight-sharing networks named In-place Ensemble Bootstrapping is proposed to improve the supernet training efficacy. In the second stage, Sandwich Gate Sparsification is proposed to assist the gate training by identifying easy and hard samples in an online way. Extensive experiments demonstrate our DS-Net consistently outperforms its static counterparts as well as state-of-the-art static and dynamic model compression methods by a large margin (up to 5.9%). Typically, DS-Net achieves 2-4× computation reduction and 1.62× real-world acceleration over ResNet-50 and MobileNet with minimal accuracy drops on ImageNet.[1]

1. Introduction

As deep neural networks are becoming deeper and wider to achieve higher performance, there is an urgent need to explore efficient models for common mobile platforms, such as self-driving cars, smartphones, drones and robots. In recent years, many different approaches have been proposed to improve the inference efficiency of neural networks, including network pruning [40, 47, 19, 20, 48, 50], weight quantization [30], knowledge distillation [2, 52, 21], manually and automatically designing of efficient networks [56, 53, 71, 51, 3, 69, 10, 16, 38, 70] and dynamic inference [4, 27, 59, 58, 41, 26, 13, 9].

Among the above approaches, dynamic inference methods have attracted increasing attention because of their promising capability of reducing computational redundancy by automatically adjusting their architecture with respect to different inputs. As illustrated in Fig. 2 right, the dynamic network learns to configure different architecture routing adaptively for each input, instead of optimizing the architecture among the whole dataset like Neural Architecture Search (NAS) or Pruning. A performance-complexity trade-

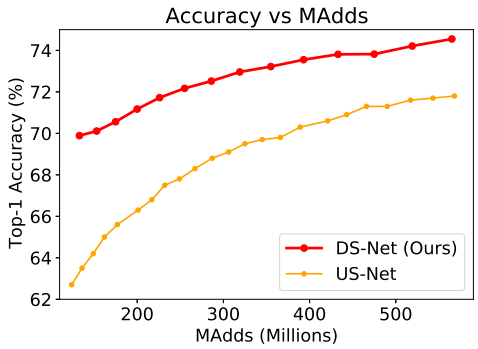

Figure 1. Universally accuracy-complexity comparison of our DS-Net and Universally Slimmable Network (US-Net) [66] (based on MobileNetV1 [23]).

Table 1. Latency comparison of ResNet-50 with 25% channels (on GeForce RTX 2080 Ti). Both *masking* and *indexing* lead to inefficient computation waste, while *slicing* achieves comparable acceleration with *ideal* (the individual ResNet-50 0.25×).

method	full	masking	indexing	slicing (ours)	ideal
latency	12.2 ms	12.4ms	16.6 ms	7.9 ms	7.2 ms

*Corresponding author.

[1]Code release: https://github.com/changlin31/DS-Net

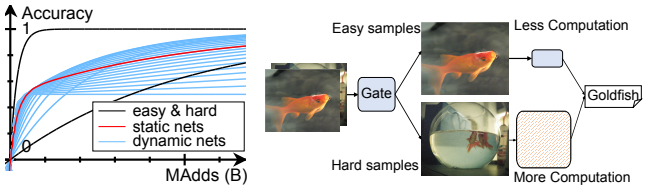

Figure 2. The motivation for designing dynamic networks to achieve efficient inference. **Left:** A simulation diagram of accuracy-complexity comparing a series of static networks (searched by NAS) with 20 dynamic inference schemes of different resource allocate proportion for easy and hard samples on a hypothetical classification dataset with evenly distributed easy and hard samples. **Right:** Illustration of dynamic networks on efficient inference. Input images are routed to use different architectures regarding their classification difficulty.

off simulated with exponential functions is shown in Fig. 2 left, the optimal solution of dynamic networks is superior to the static NAS or pruning solution. Ideally, dynamic network routing can significantly improve model performance under certain complexity constraints.

However, networks with dynamic width, *i.e.*, dynamic pruning methods [13, 26, 9], unlike its orthogonal counterparts with dynamic depth, have never achieved actual acceleration in a real-world implementation. As natural extensions of network pruning, dynamic pruning methods predictively prune the convolution filters with regard to different input at runtime. The varying sparse patterns are incompatible with computation on hardware. Actually, many of them are implemented as zero masking or inefficient path indexing, resulting in a massive gap between the theoretical analysis and the practical acceleration. As shown in Tab. 1, both masking and indexing lead to inefficient computation waste.

To address the aforementioned issues in dynamic networks, we propose **D**ynamic **S**limmable **N**etwork (DS-Net), which achieves good hardware-efficiency via dynamically adjusting filter numbers of networks at test time with respect to different inputs. To avoid the extra burden on hardware caused by dynamic sparsity, we adopt a scheme named **dynamic slicing** to keep filters static and contiguous when adjusting the network width. Specifically, we propose a **double-headed dynamic gate** with an attention head and a slimming head upon slimmable networks to predictively adjust the network width with negligible extra computation cost. The training of dynamic networks is a highly entangled bilevel optimization problem. To ensure generality of each candidate's architecture and the fairness of gate, a disentangled **two-stage training scheme** inspired by one-shot NAS is proposed to optimize the supernet and the gates separately. In the first stage, the slimmable supernet is optimized with a novel training method for weight-sharing networks, named **In-place Ensemble Bootstrapping (IEB)**. IEB trains the smaller sub-networks in the online network to fit the output logits of an ensemble of larger sub-networks in the momentum target network. Learning from the ensemble of different sub-networks will reduce the conflict among sub-networks and increase their generality. Using the exponential moving average of the online network as the momentum target network can provide a stable and accurate historical representation, and bootstrap the online network and the target network itself to achieve higher overall performance. In the second stage, to prevent dynamic gates from collapsing into static ones in the multiobjective optimization problem, a technique named **Sandwich Gate Sparsification (SGS)** is proposed to assist the gate training. During training, SGS identifies easy and hard samples online and further generates the ground truth label for the dynamic gates.

Overall, our contributions are three-fold as follows:

- We propose a new dynamic network routing regime, achieving good hardware-efficiency by predictively adjusting filter numbers of networks at test time with respect to different inputs. Unlike dynamic pruning methods, we dynamically slice the network parameters while keeping them stored statically and contiguously in hardware to prevent the extra burden of masking, indexing, and weight-copying. The dynamic routing is achieved by our proposed double-headed dynamic gate with negligible extra computation cost.
- We propose a two-stage training scheme with IEB and SGS techniques for DS-Net. Proved experimentally, IEB stabilizes the training of slimmable networks and boosts its accuracy by 1.8% and 0.6% in the slimmest and widest sub-networks respectively. Moreover, we empirically show that the SGS technique can effectively sparsify the dynamic gate and improves the final performance of DS-Net by 2%.
- Extensive experiments demonstrate our DS-Net outperforms its static counterparts [65, 66] as well as state-of-the-art static and dynamic model compression methods by a large margin (up to 5.9%, Fig. 1). Typically, DS-Net achieves 2-4× computation reduction and 1.62× real-world acceleration over ResNet-50 and MobileNet with minimal accuracy drops on ImageNet. Gate visualization proves the high dynamic diversity of DS-Net.

2. Related works

Anytime neural networks [35, 27, 24, 41, 36, 68, 66, 22] are single networks that can execute with their sub-networks under different budget constraints, thus can deploy instantly and adaptively in different application scenarios. Anytime neural networks have been studied in two orthogonal directions: networks with variable depth and variable width. **Networks with variable depth** [35, 27, 24, 41] are first studied widely, benefiting from the naturally nested structure in depth dimension and residual connections in ResNet [18] and DenseNet [28]. **Network with variable width** was first studied in [36]. Recently, slimmable networks [68, 66] using *switchable batch normalization* and *in-place distillation* achieve higher performance than their stand-alone

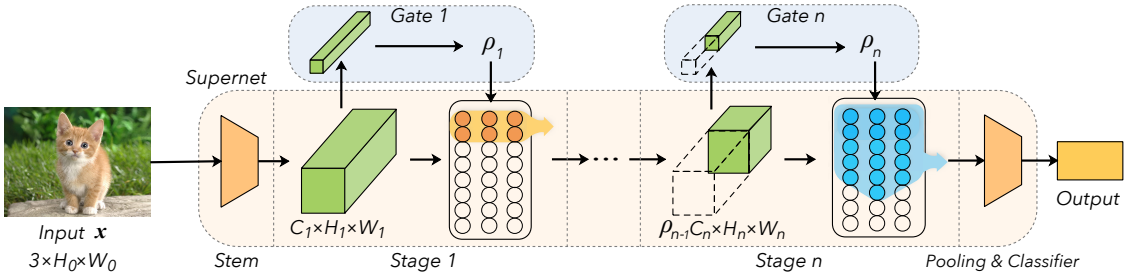

Figure 3. Architecture of DS-Net. The width of each supernet stage is adjusted adaptively by the slimming ratio ρ predicted by the gate.

counterparts in any width. Some recent works [6, 67, 22] also explore anytime neural networks in multiple dimensions, *e.g.* depth, width, kernel size, *etc.*

Dynamic neural networks [58, 59, 43, 62] change their architectures based on the input data. Dynamic networks for efficient inference aim to reduce average inference cost by using different sub-networks adaptively for inputs with diverse difficulty levels. **Networks with dynamic depth** [4, 27, 59, 58, 41] achieve efficient inference in two ways, early exiting when shallower sub-networks have high classification confidence [4, 27, 41], or skipping residual blocks adaptively [59, 58]. Recently, **dynamic pruning** methods [26, 13, 9] using a variable subset of convolution filters have been studied. Channel Gating Neural Network [26] and FBS [13] identify and skip the unimportant input channels at run-time. In GaterNet [9], a separate gater network is used to predictively select the filters of the main network. Please refer to [17] for a more comprehensive review of dynamic neural networks.

Weight sharing NAS [5, 1, 3, 45, 7, 61, 16, 38, 6, 39], aiming at designing neural network architectures automatically and efficiently, has been developing rapidly in recent two years. They integrate the whole search space of NAS into a weight sharing supernet and optimize network architecture by pursuing the best-performing sub-networks. These methods can be roughly divided into two categories: **jointly optimized methods** [45, 7, 61], in which the weight of the supernet is jointly trained with the architecture routing agent (typically a simple learnable factor for each candidate route); and **one-shot methods** [5, 1, 3, 16, 38, 6, 39], in which the training of the supernet parameters and architecture routing agent are disentangled. After fair and sufficient training, the agent is optimized with the weights of supernet frozen.

3. Dynamic Slimmable Network

Our dynamic slimmable network achieves dynamic routing for different samples by learning a slimmable supernet and a dynamic gating mechanism. As illustrated in Fig. 3, the supernet in DS-Net refers to the whole module undertaking the main task. In contrast, the dynamic gates are a series of predictive modules that route the input to use sub-networks with different widths in each stage of the supernet.

In previous dynamic networks [58, 59, 43, 62, 4, 27, 41, 26, 13, 9], the dynamic routing agent and the main network are jointly trained, analogous to jointly optimized NAS methods [45, 7, 61]. Inspired by one-shot NAS methods [5, 1, 3, 16, 38], we propose a disentangled two-stage training scheme to ensure the generality of every path in our DS-Net. In **Stage I**, we disable the slimming gate and train the supernet with the IEB technique, then in **Stage II**, we fix the weights of the supernet and train the slimming gate with the SGS technique.

3.1. Dynamic Supernet

In this section, we first introduce the hardware efficient channel slicing scheme and our designed supernet, then present the IEB technique and details of training Stage I.

Supernet and Dynamic Channel Slicing. In some of dynamic networks, such as dynamic pruning [26, 13] and conditional convolution [62, 42], the convolution filters $\mathcal{W}$ are conditionally parameterized by a function $\mathcal{A}(\theta, \mathcal{X})$ to the input $\mathcal{X}$. Generally, the dynamic convolution has a form of:

$$\mathcal{Y} = \mathcal{W}_{\mathcal{A}(\theta,\mathcal{X})} * \mathcal{X}, \tag{1}$$

where $\mathcal{W}_{\mathcal{A}(\theta,\mathcal{X})}$ represents the selected or generated input-dependent convolution filters. Here $*$ is used to denote a matrix multiplication. Previous dynamic pruning methods [26, 13] reduce theoretical computation cost by varying the channel sparsity pattern according to the input. However, they fail to achieve real-world acceleration because their hardware-incompatible channel sparsity results in repeatedly indexing and copying selected filters to a new contiguous memory for multiplication. To achieve practical acceleration, filters should remain contiguous and relatively static during dynamic weight selection. Base on this analysis, we design a architecture routing agent $\mathcal{A}(\theta)$ with the inductive bias of always outputting a *dense* architecture, *e.g.* a *slice-able* architecture. Specifically, we consider a convolutional layer with at most N output filters and M input channels. Omitting the spatial dimension, its filters can be denoted as $\mathbf{W} \in \mathbb{R}^{N\times M}$. The output of the architecture routing agent $\mathcal{A}(\theta)$ for this convolution would be a slimming ratio $\rho \in (0, 1]$ indicating that the first **piece-wise** $\rho \times N$ of the output filters are selected. Then, a dynamic slice-able convolution is defined as follows:

$$\mathcal{Y} = \mathbf{W}[\,:\rho\times N] * \mathcal{X}, \tag{2}$$

where $[\,:\,]$ is a slice operation denoted in a python-like style. **Remarkably**, the slice operation $[\,:\,]$ and the dense matrix multiplication $*$ are much more efficient than an indexing operation or a sparse matrix multiplication in real-world

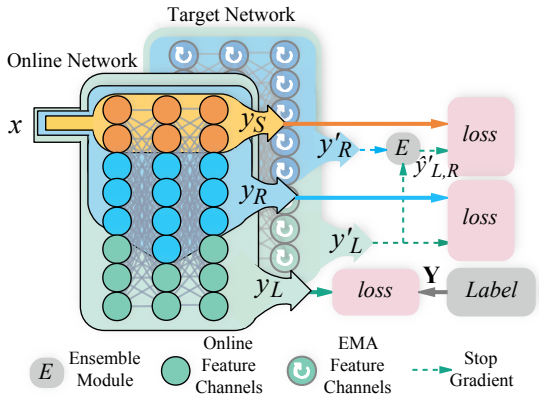

Figure 4. Training process of slimmable supernet with In-place Ensemble Bootstrapping.

implementation, which guarantees a practical acceleration of using our slice-able convolution.

After aggregating the slice-able convolutions sequentially, a supernet executable at different widths is formed. Paths with different widths can be seen as sub-networks. By disabling the routing agent, the supernet is analogous to a slimmable network [68, 66], and can be trained similarly.

In-place Ensemble Bootstrapping. The *sandwich rule* and *in-place distillation* techniques [66] proposed for Universally Slimmable Networks enhanced their overall performance. In *in-place distillation*, the widest sub-network is used as the target network generating soft labels for other sub-networks. However, acute fluctuation appeared in the weight of the widest sub-network can cause convergence hardship, especially in the early stage of training. As observed in BigNAS [67], training a more complex model with in-place distillation could be highly unstable. Without residual connection and special weight initialization tricks, the loss exploded at the early stage and can never converge. To overcome the convergence hardship in slimmable networks and improve the overall performance of our supernet, we proposed a training scheme named *In-place Ensemble Bootstrapping (IEB)*.

In recent years, a growing number of self-supervised methods with bootstrapping [14, 15, 8] and semi-supervised methods based on consistency regularization [34, 57] use their historical representations to produce targets for the online network. Inspired by this, we propose to bootstrap on previous representations in our *supervised* in-place distillation training. We use the exponential moving average (EMA) of the model as the target network that generates soft labels. Let θ and θ' denote the parameters of the online network and the target network, respectively. We have:

$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t, \tag{3}$$

where α is a momentum factor controlling the ratio of the historical parameter and t is a training timestamp which is usually measured by a training iteration. During training, the EMA of the model are more stable and more precise than the online network, thus can provide high quality target for the slimmer sub-networks.

As pointed out in [54, 55], an ensemble of teacher networks can generate more diverse, more accurate and more general soft labels for distillation training of the student network. In our supernet, there are tons of sub-models with different architectures, which can generate different soft labels. Motivated by this, we use different sub-networks as a teacher ensemble when performing in-place distillation. The overall train process is shown in Fig. 4. Following the sandwich rule [66], the widest (denoted with L), the slimmest (denoted with S) and n random width sub-networks (denoted with R) are sampled in each training step. Sub-network at the largest width is trained to predict the ground truth label $\mathbf{Y}$; n sub-networks with random width are trained to predict the soft label generated by the widest sub-network of the target network, $\mathcal{Y}'_L(\theta')$; the slimmest sub-network is trained to predict the probability ensemble of all the aforementioned sub-networks in the target network:

$$\widehat{\mathcal{Y}'}_{L,R}(\theta') = \frac{1}{n+1}\left(\mathcal{Y}'_L(\theta') + \sum_{i=1}^{n}\mathcal{Y}'_R(\theta')\right). \tag{4}$$

To sum up, the IEB losses for the supernet training are:

$$\begin{cases} \mathcal{L}_L^{IEB}(\theta) = \mathcal{L}_{CE}(\mathcal{Y}_L(\theta), \mathbf{Y}), \\ \mathcal{L}_R^{IEB}(\theta) = \mathcal{L}_{CE}(\mathcal{Y}_R(\theta), \mathcal{Y}'_L(\theta')), \\ \mathcal{L}_S^{IEB}(\theta) = \mathcal{L}_{CE}(\mathcal{Y}_S(\theta), \widehat{\mathcal{Y}'}_{L,R}(\theta')), \end{cases} \tag{5}$$

3.2. Dynamic Slimming Gate

In this section, we design the channel gate function $\mathcal{A}(\theta, \mathcal{X})$ that generates the factor ρ in Eqn. (2) and present the *double-headed design* of the dynamic gate. Then, we introduce the details of training stage II with an advanced technique that is *sandwich gate sparsification (SGS)*.

Double-headed Design. There are two possible ways to transform a feature map into a slimming ratio ρ in Eqn. (2): **(i) scalar design** directly output a sigmoid activated scalar ranging from 0 to 1 to be the slimming ratio; **(ii) one-hot design** use an `argmax`/`softmax` activated one-hot vector to choose the respective slimming ratio ρ in a discrete candidate list vector L_ρ. Both of the implementations are evaluated and compared in Sec. 4.4. Here, we thoroughly describe our dynamic slimming gate with the better-performing **one-hot design**. To reduce the input feature map $\mathcal{X}$ to a one-hot vector, we divide $\mathcal{A}(\theta, \mathcal{X})$ to two functions:

$$\mathcal{A}(\theta, \mathcal{X}) = \mathcal{F}(\mathcal{E}(\mathcal{X})), \tag{6}$$

where $\mathcal{E}$ is an encoder that reduces feature maps to a vector and the function $\mathcal{F}$ maps the reduced feature to a one-hot vector used for the subsequent channel slicing. Considering the n-th gate in Fig. 3, given a input feature $\mathcal{X}$ with dimension $\rho_{n-1}C_n \times H_n \times W_n$, $\mathcal{E}(\mathcal{X})$ reduces it to a vector $\mathcal{X}_\mathcal{E} \in \mathbb{R}^{\rho_{n-1}C_n}$ which can be further mapped to a one-hot vector. By computing the dot product of this one-hot vector and L_ρ, we have the newly predicted slimming ratio:

$$\rho_n = \mathcal{A}(\theta, \mathcal{X}) \cdot L_\rho. \tag{7}$$

Similar to prior works [25, 64] on channel attention and gating, we simply utilize average pooling as a light-weight

encoder $\mathcal{E}$ to integrate spatial information. As for feature mapping function $\mathcal{F}$, we adopt two fully connected layers with weights $\mathbf{W}_1 \in \mathbb{R}^{d \times C_n}$ and $\mathbf{W}_2 \in \mathbb{R}^{g \times d}$ (where d represents the hidden dimension and g represents the number of candidate slimming ratio) and a ReLU non-linearity layer σ in between to predict scores for each slimming ratio choice. An `argmax` function is subsequently applied to generate a one-hot vector indicating the predicted choice:

$$\mathcal{F}(\mathcal{X}_{\mathcal{E}}) = \texttt{argmax}(\mathbf{W}_2(\sigma(\mathbf{W}_1[:, : \rho_{n-1}C_n](\mathcal{X}_{\mathcal{E}})))). \quad (8)$$

Note that input $\mathcal{X}$ with dynamic channel number $\rho \times C$ is projected to a vector with fixed length by the dynamically sliced weight $\mathbf{W}_1[:, : \rho_{n-1}C_n]$.

Our proposed channel gating function has a similar form with recent channel attention methods [25, 64]. The attention mechanism can be integrated into our gate with nearly zero cost, by adding another fully connected layer with weights $\mathbf{W}_3$ that projects the hidden vector back to the original channel number $\rho_{n-1}C_n$. Based on the conception above, we propose a *double-headed dynamic gate* with a soft channel attention head and a hard channel slimming head.The channel attention head can be defined as follows:

$$\hat{\mathcal{X}} = \mathcal{X} * \delta(\mathbf{W}_3[: \rho_{n-1}C_n, :](\sigma(\mathbf{W}_1[:, : \rho_{n-1}C_n](\mathcal{X})))), \quad (9)$$

where $\delta(x) = 1 + \texttt{tanh}(x)$ is the activation function adopted for the attention head. Unlike the slimming head, the channel attention head is activated in training stage I.

Sandwich Gate Sparsification. In training stage II, we propose to use the end-to-end classification cross-entropy loss $\mathcal{L}_{cls}$ and a complexity penalty loss $\mathcal{L}_{cplx}$ to train the gate, aiming to choose the most efficient and effective sub-networks for each instance. To optimize the non-differentiable slimming head of dynamic gate with $\mathcal{L}_{cls}$, we use `gumbel-softmax` [31], a classical way to optimize neural networks with `argmax` by relaxing it to differentiable `softmax` in gradient computation.

However, we empirically found that the gate easily collapses into a static one even if we add Gumbel noise [31] to help the optimization of `gumbel-softmax`. Apparently, using only `gumbel-softmax` technique is not enough for this multi-objective dynamic gate training. To further overcome the convergence hardship and increase the dynamic diversity of the gate, a technique named *Sandwich Gate Sparsification (SGS)* is further proposed. We use the slimmest sub-network and the whole network to identify easy and hard samples online and further generate the ground truth slimming factors for the slimming heads of all the dynamic gates.

As analysed in [66], wider sub-networks should always be more accurate because the accuracy of slimmer ones can always be achieved by learning new connections to zeros. Thus, given a well-trained supernet, input samples can be roughly classified into three difficulty levels: **a) Easy samples** $\mathcal{X}_{easy}$ that can be correctly classified by the slimmest sub-network; **b) Hard samples** $\mathcal{X}_{hard}$ that can not be correctly classified by the widest sub-network; **c) Dependent samples** $\mathcal{X}_{dep}$: Other samples in between. In order to minimize the computation cost, easy samples should always be routed to the slimmest sub-network (*i.e.* gate target $\mathcal{T}(\mathcal{X}_{easy}) = [1, 0, \ldots, 0]$). For dependent samples and hard samples, we always encourage them to pass through the widest sub-network, even if the hard samples can not be correctly classified (*i.e.* $\mathcal{T}(\mathcal{X}_{hard}) = \mathcal{T}(\mathcal{X}_{dep}) = [0, \ldots, 0, 1]$). Another gate target strategy is also discussed in Sec. 4.4.

Based on the generated gate target, we define the SGS loss that facilitates the gate training:

$$\begin{aligned}\mathcal{L}_{SGS} = \mathbb{T}_{slim}(\mathcal{X}) * \mathcal{L}_{CE}(\mathcal{X}, \mathcal{T}(\mathcal{X}_{easy})) \\ + (\neg\mathbb{T}_{slim}(\mathcal{X})) * \mathcal{L}_{CE}(\mathcal{X}, \mathcal{T}(\mathcal{X}_{hard}))\end{aligned} \quad (10)$$

where $\mathbb{T}_{slim}(\mathcal{X}) \in \{0, 1\}$ represents whether $\mathcal{X}$ is truely predicted by the slimmest sub-network and $\mathcal{L}_{CE}(\mathcal{X}, \mathcal{T}) = -\sum \mathcal{T} * \log(\mathcal{X})$ is the Cross-Entropy loss over `softmax` activated gate scores and the generated gate target.

4. Experiments

Dataset. We evaluate our method on two classification datasets (*i.e.*, ImageNet [11] and CIFAR-10 [33]) and a standard object detection dataset (*i.e.*, PASCAL VOC [12]). The ImageNet dataset is a large-scale dataset containing 1.2 M `train` set images and 50 K `val` set images in 1000 classes. We use all the training data in both of the two training stages. Our results are obtained on the `val` set with image size of 224×224. We also test the transferability of our DS-Net on CIFAR-10, which comprises 10 classes with 50,000 training and 10,000 test images. Note that few previous works on dynamic networks and network pruning reported results on object detection. We take PASCAL VOC, one of the standard datasets for evaluating object detection performance, as an example to further test the generality of our dynamic networks on object detection. All the detection models are trained with the combined dataset from 2007 `trainval` and 2012 `trainval` and tested on VOC 2007 `test` set.

Architecture details. Following previous works on static and dynamic network pruning, we use two representative networks, *i.e.*, the heavy residual network ResNet 50 [18] and the lightweight non-residual network MobileNetV1 [23], to evaluate our method.

In Dynamic Slimmable ResNet 50 **(DS-ResNet)**, we insert our double-headed gate in the begining of each residual blocks. The slimming head is only used in the first block of each stage. Each one of those blocks contains a skip connection with a projection layer, *i.e.* 1×1 convolution. The filter number of this projection convolution is also controlled by the gate to avoid channel inconsistency when adding skip features with residual output. In other residual blocks, the slimming heads of the gates are disabled and all the layers in those blocks inherit the widths of the first blocks of each stage. To sum up, there are 4 gates (one for each stage) with both heads enabled. Every gates have 4 equispaced candidate slimming ratios, *i.e.* $\rho \in \{0.25, 0.5, 0.75, 1\}$. The total

routing space contains $4^4 = 256$ possible paths with different computation complexities. All batch normalization (BN) layers in DS-ResNet are replaced with group normalization to avoid test-time representation shift caused by inaccurate BN statistics in weight-sharing networks [68, 66].

Unlike DS-ResNet, we only use one single slimming gate after the fifth depthwise separable convolution block of Dynamic Slimmable MobileNetV1 **(DS-MBNet)**. Specifically, a fixed slimming ratio $\rho = 0.5$ is used in the first 5 blocks while the width of the rest 8 blocks are controlled by the gate with the candidate slimming ratios $\rho \in [0.35 : 0.05 : 1.25]$. This architecture with only 18 paths in its routing space is similar to an uniform slimmable network [68, 66], guaranteeing itself the practicality to use batch normalization. Following [66], we perform BN recalibration for all the 18 paths in DS-MBNet after the supernet training stage.

Training details. We train our supernet with 512 total batch size on ImageNet, using SGD optimizer with 0.2 initial learning rate for DS-ResNet and 0.08 initial learning rate for DS-MBNet, respectively. We use cosine learning rate scheduler to reduce the learning rate to its 1% in 150 epochs. Other settings are following previous works on slimmable networks [68, 66, 65]. For gate training, we use SGD optimizer with 0.05 initial learning rate for a total batch size of 512. The learning rate decays to 0.9× of its value in every epoch. It takes 10 epochs for the gate to converge. For transfer learning experiments on CIFAR-10, we follow similar settings with [32] and [29]. We transfer our supernet for 70 epochs including 15 warm-up epochs and use cosine learning rate scheduler with an initial learning rate of 0.7 for a total batch size of 1024. For object detection task , we train all the networks following [46] and [44] with a total batch size of 128 for 300 epochs. The learning rate is set to 0.004 at the first, then divided by 10 at epoch 200 and 250.

4.1. Main Results on ImageNet

We first validate the effectiveness of our method on ImageNet. As shown in Tab. 2 and Fig. 5, DS-Net with different computation complexity consistently outperforms recent static pruning methods, dynamic inference methods and NAS methods. **First**, our DS-ResNet and DS-MBNet models achieve 2-4× computation reduction over ResNet-50 (76.1% [18]) and MobileNetV1 (70.9% [23]) with minimal accuracy drops (0% to -1.5% for ResNet and +0.9% to -0.8% for MobileNet). We also tested the real world latency on efficient networks. Compare to the ideal acceleration tested on channel scaled MobileNetV1, which is 1.31× and 1.91×, our DS-MBNet achieves comparable 1.17× and 1.62× acceleration with *much* higher performance. In paticular, our DS-MBNet surpasses the original and the channel scaled MobileNetV1 [23] by **3.6%**, **4.4%** and **6.8%** with similar MAdds and minor increase in latency. **Second**, our method outperforms classic and state-of-the-art static pruning methods in a large range. Remarkably, DS-MBNet outperforms

Table 2. Comparison of state-of-the-art efficient inference methods on ImageNet. Brown denotes network pruning methods, Blue denotes dynamic inference methods, Orange denotes architecture search methods and Purple denotes our method.

	Method	MAdds	Latency	Top-1 Acc.
3B MAdds	SFP [19]	2.9B	-	75.1
	ThiNet-70 [50, 49]	2.9B	-	75.8
	MetaPruning 0.85 [48]	3.0B	-	76.2
	ConvNet-AIG-50 [58]	3.1B	-	76.2
	AutoSlim [65]	3.0B	-	76.0
	DS-ResNet-L (Ours)	3.1B	-	**76.6**
2B MAdds	ResNet-50 0.75× [18]	2.3B	-	74.9
	S-ResNet-50 [68]	2.3B	-	74.9
	ThiNet-50 [50, 49]	2.1B	-	74.7
	CP [20]	2.0B	-	73.3
	MetaPruning 0.75 [48]	2.0B	-	75.4
	MSDNet [27]	2.0B	-	75.5
	AutoSlim [65]	2.0B	-	75.6
	DS-ResNet-M (Ours)	2.2B	-	**76.1**
1B MAdds	ResNet-50 0.5× [18]	1.1B	-	72.1
	ThiNet-30 [50, 49]	1.2B	-	72.1
	MetaPruning 0.5 [48]	1.0B	-	73.4
	GFNet [60]	1.2B	-	73.8
	DS-ResNet-S (Ours)	1.2B	-	**74.6**
500M MAdds	MBNetV1 1.0× [23]	569M	63ms	70.9
	US-MBNetV1 1.0× [66]	569M	-	71.8
	AutoSlim [65]	572M	-	73.0
	DS-MBNet-L (Ours)	565M	69ms	**74.5**
300M MAdds	MBNetV1 0.75× [23]	317M	48ms	68.4
	US-MBNetV1 0.75× [66]	317M	-	69.5
	NetAdapt [63]	284M	-	69.1
	Meta-Pruning [48]	281M	-	70.6
	EagleEye [37]	284M	-	70.9
	CG-Net-A [26]	303M	-	70.3
	AutoSlim [65]	325M	-	71.5
	DS-MBNet-M (Ours)	319M	54ms	**72.8**
150M MAdds	MBNetV1 0.5× [23]	150M	33ms	63.3
	US-MBNetV1 0.5× [66]	150M	-	64.2
	AutoSlim [65]	150M	-	67.9
	DS-MBNet-S (Ours)	153M	39ms	**70.1**

the SOTA pruning methods EagleEye [37] and Meta-Pruning [48] by **1.9%** and **2.2%**. **Third**, our DS-Net maintain superiority comparing with powerful dynamic inference methods with varying depth, width or input resolution. For example, our DS-MBNet-M surpasses dynamic pruning method CG-Net [26] by **2.5%**. **Fourth**, our DS-Net also consistently outperforms its static counterparts. Our DS-MBNet-S surpasses AutoSlim [65] and US-Net [66] by **2.2%** and **5.9%**.

4.2. Transferability

To evaluate the transferability of DS-Net and its dynamic gate, we perform transfer learning in two settings: **(i) DS-Net w/o gate transfer**: we transfer the supernet without slimming gate to CIFAR-10 and retrain the dynamic gate. **(ii) DS-Net w/ gate transfer**: we first transfer the supernet then load the ImageNet trained gate and perform transfer leaning for the gate. The results along with the transfer learning results of the original ResNets are shown in Tab. 3. Gate transfer boosts the performance of DS-ResNet by 0.4% on CIFAR-10, demonstrating the transferability of dynamic gate. **Remarkably**, both of our transferred DS-ResNet outperforms the original ResNet-50 in a large range (0.6% and

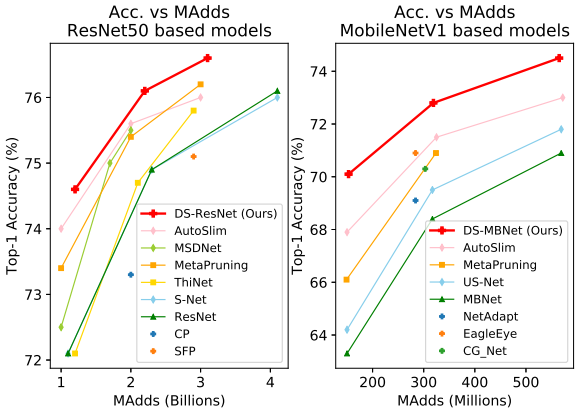

Figure 5. Accuracy *vs.* complexity on ImageNet.

Table 3. Comparison of transfer learning performance on CIFAR-10. GT stands for gate transfer.

Model	MAdds	Top-1 Acc.
ResNet-50 [18, 32]	4.1B	96.8
ResNet-101 [18, 32]	7.8B	97.6
DS-ResNet w/o GT	1.7B	97.4
DS-ResNet w/ GT	1.6B	97.8

Table 4. Performance comparision of DS-MBNet and MobileNet with FSSD on VOC object detection task.

Model	MAdds	mAP
MBNetV1 + FSSD [23, 44]	4.3B	71.9
DS-MBNet-S + FSSD	2.3B	70.7
DS-MBNet-M + FSSD	2.7B	72.8
DS-MBNet-L + FSSD	3.2B	**73.7**

1.0%) with about 2.5 × computation reduction. Among them, DS-ResNet with gate transfer even outperforms the larger ResNet-101 with 4.9× fewer computation complexity, proving the superiority of DS-Net in transfer learning.

4.3. Object Detection

In this section, we evaluate and compare the performance of original MobileNet and DS-MBNet used as feature extractor in object detection with Feature Fusion Single Shot Multibox Detector(FSSD) [44]. We use the features from the 5-th, 11-th and 13-th depthwise convolution blocks (with the output stride of 8, 16, 32) of MobileNet for the detector. When using DS-MBNet as the backbone, all the features from dynamic source layers are projected to a fixed channel dimention by the feature transform module in FSSD [44].

Results on VOC 2007 `test` set are given in Tab. 4. Comparing to MobileNetV1, DS-MBNet-M and DS-MBNet-L with FSSD achieves 0.9 and 1.8 mAP improvement with 1.59× and 1.34× computation reduction respectively, which demonstrates that our DS-Net remain its superiority after deployed as the backbone network in object detection task.

4.4. Ablation study

In-place Ensemble Bootstrapping. We statistically analysis the effect of IEB technique with MobileNetV1. We train a Slimmable MobileNetV1 supernet with three settings: original in-place distillation, in-place distillation with

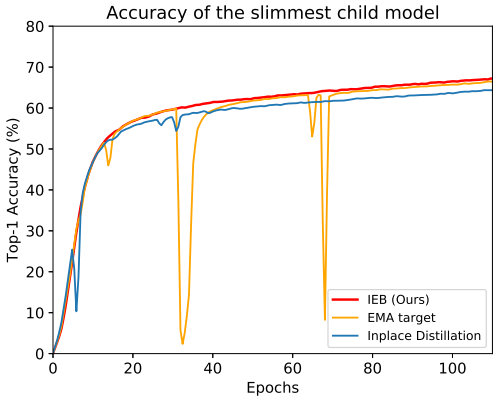

Figure 6. Evaluation accuracy of the slimmest sub-network during supernet training with three different training schemes.

Table 5. Ablation analysis of In-place Ensemble Bootstrapping.

EMA	Ensemble	slimmest	widest
		66.5	74.0
✓		68.1	74.3
✓	✓	68.3	74.6

EMA target and our complete IEB technique. As shown in Tab. 5, the slimmest and widest sub-network trained with EMA target surpassed the baseline by 1.6% and 0.3% respectively. With IEB, the supernet improves 1.8% and 0.6% on its slimmest and widest sub-networks comparing with in-place distillation. The evaluation accuracy progression curves of the slimmest sub-networks trained with these three settings are illustrated in Fig. 6. The beginning stage of in-place distillation is unstable. Adopting EMA target improves the performance. However, there are a few sudden drops of accuracy in the middle of the training with EMA target. Though being able to recover in several epochs, the model may still be potentially harmed by those fluctuation. After fully adopting IEB, the model converges to a higher final accuracy without any conspicuous fluctuations in the training process, demonstrating the effectiveness of our IEB technique in stablizing the training and boosting the overall performance of slimmable networks.

Effect of losses. To examine the impact of the three losses used in our gate training, *i.e. target loss* $\mathcal{L}_{cls}$, *complexity loss* $\mathcal{L}_{cplx}$ and *SGS loss* $\mathcal{L}_{SGS}$, we conduct extensive experiments with DS-ResNet on ImageNet, and summarize the results in Tab. 6 and Fig. 7 left. Firstly, as illustrated in Fig. 7 left, models trained with SGS (red line) are more efficient than models trained without it (purple line). Secondly, as shown in Tab. 6, with target loss, the model pursues better performance while ignoring computation cost; complexity loss pushes the model to be lightweight while ignoring the performance; SGS loss itself can achieve a balanced complexity-accuracy trade-off by encouraging easy and hard samples to use slim and wide sub-networks, respectively.

SGS strategy. Though we always want the easy samples to be routed to the slimmest sub-network, there are two possible

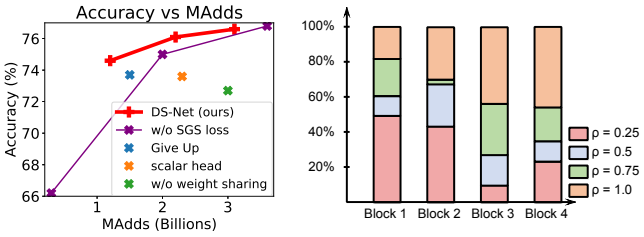

Figure 7. **(Left)** Illustration of accuracy *vs.* complexity of models in Tab. 6 and Tab. 7. **(Right)** Gate distribution of DS-ResNet-M. The height of those colored blocks illustrate the partition of input samples that are routed to the sub-networks with respective slimming ratio ρ.

Table 6. Ablation analysis of losses on ImageNet. Results in bold that use SGS loss achieve good performance-complexity trade-off.

Target	Complexity	SGS	MAdds	Top-1 Acc.
✓			3.6B	76.8
	✓		0.3B	66.2
		✓ Give Up	1.5B	73.7
		✓ Try Best	**3.1B**	**76.6**
✓	✓		2.0B	75.0
	✓	✓ Try Best	**1.2B**	**74.6**
✓	✓	✓ Try Best	**2.2B**	**76.1**

Table 7. Ablation analysis of gate design on DS-ResNet.

weight sharing	slimming head	MAdds	Top-1 Acc.
✓	scalar	2.3B	73.6
	one-hot	3.0B	72.7
✓	one-hot	3.1B	76.6

target definition for hard samples in SGS loss: **(i) Try Best:** Encourage the hard samples to pass through the widest sub-network, even if they can not be correctly classified (*i.e.* $\mathcal{T}(\mathcal{X}_{hard}) = [0, \ldots, 0, 1]$). **(ii) Give Up:** Push the hard samples to use the slimmest path to save computation cost (*i.e.* $\mathcal{T}(\mathcal{X}_{hard}) = [1, 0, \ldots, 0]$). In both of the strategies, dependent samples are encouraged to use the widest sub-network (*i.e.* $\mathcal{T}(\mathcal{X}_{dependent}) = [0, \ldots, 0, 1]$). The results for both of the strategies are shown in Tab. 6 and Fig. 7 left. As shown in the third and fourth lines in Tab. 6, *Give Up* strategy lowers the computation complexity of the DS-ResNet but greatly harms the model performance. The models trained with *Try Best* strategy (red line in Fig. 7 left) outperform the one trained with *Give Up* strategy (blue dot in Fig. 7 left) in terms of efficiency. This can be attribute to *Give Up* strategy's optimization difficulty and the lack of samples that targeting on the widest path (dependent samples only account for about 10% of the total training samples). These results prove our *Try Best* strategy is easier to optimize and can generalize better on validation set or new data.

Gate design. **First**, to evaluate the effect of our weight-sharing double-headed gate design, we train a DS-ResNet without sharing the the first fully-connected layer for comparison with SGS loss only. As shown in Tab. 7 and Fig. 7 left, the performance of DS-ResNet increase substantially (3.9%) by applying the weight sharing design (green dot *vs.* red line in Fig. 7 left). This might be attribute to overfitting of the slimming head. As observed in our experiment, sharing the first fully-connected layer with attention head can greatly improve the generality. **Second**, we also trained a DS-ResNet with *scalar design* (refer to Sec 3.2) of the slimming head to compare with *one-hot design*. Both of the networks are trained with SGS loss only. The results are present in Tab. 7 and Fig. 7 left. The performance of *scalar design* (orange dot in Fig. 7 left) is much lower than the *one-hot design* (red line in Fig. 7 left), indicating that the scalar gate could not route the input to the correct paths.

4.5. Gate visualization

To demonstrate the dynamic diversity of our DS-Net, we visualize the gate distribution of DS-ResNet over the validation set of ImageNet in Fig. 7 right. In block 1 and 2, about half of the inputs are routed to the slimmest sub-network with 0.25 slimming ratio, while in higher level blocks, about half of the inputs are routed to the widest sub-network. For all the gate, the slimming ratio choices are highly input-dependent, demonstrating the high dynamic diversity of our DS-Net.

5. Conclusion

In this paper, we have proposed Dynamic Slimmable Network (DS-Net), a novel dynamic network on efficient inference, achieving good hardware-efficiency by predictively adjusting the filter numbers at test time with respect to different inputs. We propose a two stage training scheme with In-place Ensemble Bootstrapping (IEB) and Sandwich Gate Sparsification (SGS) technique to optimize DS-Net. We demonstrate that DS-Net can achieve 2-4$\times$ computation reduction and 1.62$\times$ real-world acceleration over ResNet-50 and MobileNet with minimal accuracy drops on ImageNet. Proved empirically, DS-Net and can surpass its static counterparts as well as state-of-the-art static and dynamic model compression method on ImageNet by a large margin (>2%) and can generalize well on CIFAR-10 classification task and VOC object detection task.

Acknowledgments

This work was supported in part by National Key R&D Program of China under Grant No. 2020AAA0109700, National Natural Science Foundation of China (NSFC) under Grant No.U19A2073, No.61976233 and No.61906109, Guangdong Province Basic and Applied Basic Research (Regional Joint Fund-Key) Grant No.2019B1515120039, Shenzhen Outstanding Youth Research Project (Project No. RCYX20200714114642083), Shenzhen Basic Research Project (Project No. JCYJ20190807154211365), Leading Innovation Team of the Zhejiang Province (2018R01017) and CSIG Young Fellow Support Fund. Dr Xiaojun Chang is partially supported by the Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA) (DE190100626).

References

[1] Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *ICML*, 2019. 3

[2] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NeurIPS*, 2014. 1

[3] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018. 1, 3

[4] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for fast test-time prediction. *arXiv:1702.07811*, 2017. 1, 3

[5] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. SMASH: one-shot model architecture search through hypernetworks. In *ICLR*, 2018. 3

[6] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020. 3

[7] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 3

[8] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 4

[9] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. You look twice: Gaternet for dynamic filter selection in cnns. In *CVPR*, 2019. 1, 2, 3

[10] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Hongdong Li, Tom Drummond, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. In *NeurIPS*, 2020. 1

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[12] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 5

[13] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. *arXiv:1810.05331*, 2018. 1, 2, 3

[14] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020. 4

[15] Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *ICML*, 2020. 4

[16] Zichao Guo, X. Zhang, H. Mu, Wen Heng, Z. Liu, Y. Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020. 1, 3

[17] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *arXiv:2102.04906*, 2021. 3

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 5, 6, 7

[19] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI*, 2018. 1, 6

[20] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 1, 6

[21] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015. 1

[22] Lu Hou, Lifeng Shang, Xin Jiang, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. In *NeurIPS*, 2020. 2, 3

[23] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017. 1, 5, 6, 7

[24] Hanzhang Hu, Debadeepta Dey, Martial Hebert, and J Andrew Bagnell. Learning anytime predictions in neural networks via adaptive loss balancing. In *AAAI*, 2019. 2

[25] J Hu, L Shen, S Albanie, G Sun, and E Wu. Squeeze-and-excitation networks. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 4, 5

[26] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. In *NeurIPS*, 2019. 1, 2, 3, 6

[27] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018. 1, 2, 3, 6

[28] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 2

[29] Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *NeurIPS*, 2019. 6

[30] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, 2018. 1

[31] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017. 5

[32] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *CVPR*, 2018. 6, 7

[33] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 5

[34] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv:1610.02242*, 2016. 4

[35] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2017. 2

[36] Hankook Lee and Jinwoo Shin. Anytime neural prediction via slicing networks vertically. *arXiv:1807.02609*, 2018. 2

[37] Bailin Li, Bowen Wu, Jiang Su, Guangrun Wang, and Liang Lin. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In *ECCV*, 2020. 6

[38] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *CVPR*, 2020. 1, 3

[39] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. *arXiv:2103.12424*, 2021. 3

[40] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv:1608.08710*, 2016. 1

[41] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *ICCV*, 2019. 1, 2, 3

[42] Yunsheng Li, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Ye Yu, Lu Yuan, Zicheng Liu, Mei Chen, and Nuno Vasconcelos. Revisiting dynamic convolution via matrix decomposition. *arXiv:2103.08756*, 2021. 3

[43] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *CVPR*, 2020. 3

[44] Zuoxin Li and Fuqiang Zhou. Fssd: feature fusion single shot multibox detector. *arXiv:1712.00960*, 2017. 6, 7

[45] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*, 2019. 3

[46] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 6

[47] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 1

[48] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *ICCV*, 2019. 1, 6

[49] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *ICLR*, 2019. 6

[50] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017. 1, 6

[51] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys*, 2021. 1

[52] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015. 1

[53] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 1

[54] Zhiqiang Shen, Zhankui He, and Xiangyang Xue. Meal: Multi-model ensemble via adversarial learning. In *AAAI*, 2019. 4

[55] Zhiqiang Shen and Marios Savvides. Meal v2: Boosting vanilla resnet-50 to 80%+ top-1 accuracy on imagenet without tricks. *arXiv:2009.08453*, 2020. 4

[56] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 1

[57] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 4

[58] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018. 1, 3, 6

[59] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, 2018. 1, 3

[60] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. In *NeurIPS*, 2020. 6

[61] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 2019. 3

[62] Brandon Yang, Gabriel Bender, Quoc V. Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *NeurIPS*, 2019. 3

[63] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *ECCV*, 2018. 6

[64] Zongxin Yang, Linchao Zhu, Yu Wu, and Yi Yang. Gated channel transformation for visual recognition. In *CVPR*, 2020. 4, 5

[65] Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv:1903.11728*, 2019. 2, 6

[66] Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In *ICCV*, 2019. 1, 2, 4, 5, 6

[67] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, T. Huang, Xiaodan Song, and Quoc V. Le. Bignas: Scaling up neural architecture search with big single-stage models. In *ECCV*, 2020. 3, 4

[68] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas S. Huang. Slimmable neural networks. In *ICLR*, 2019. 2, 4, 6

[69] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, Zongyuan Ge, and Steven W. Su. Differentiable neural architecture search in equivalent space with exploration enhancement. In *NeurIPS*, 2020. 1

[70] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, and Steven W. Su. Overcoming multi-model forgetting in one-shot NAS with diversity maximization. In *CVPR*, 2020. 1

[71] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, Chuan Zhou, Zongyuan Ge, and Steven W Su. One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1