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. 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 [43, 50, 21, 22, 51, 53], weight quantization [33], knowledge distillation [2, 56, 23], manually and automatically designing of efficient networks [60, 57, 77, 55, 3, 75, 10, 18, 41, 76] and dynamic inference [4, 30, 64, 63, 44, 29, 14].

Among the above approaches, dynamic inference methods, including networks with dynamic depth [4, 30, 64, 63, 44] and dynamic width [14, 29, 9] have attracted increasing attention because of their promising capability of reducing computational redundancy by automatically adjusting their architecture for different inputs. As illustrated in Fig. 2, 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
Accuracy which achieves good hardware-efficiency via dynamically works, we propose $D$ with computation on hardware. Actually, many of them are to keep filters static and contiguous when dynamic slicing. To address the aforementioned issues in dynamic networks, we propose Dynamic Slimmable Network (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.

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 [71, 72] 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 [38, 30, 27, 44, 39, 74, 72, 24] 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** [38, 30, 27, 44] are first studied widely, benefiting from the naturally nested structure in depth dimension and residual connections in ResNet [20] and DenseNet [31]. **Network with variable width** was first studied in [39]. Recently, slimmable networks [74, 72] using switchable batch normalization and in-place distillation achieve higher performance than their stand-alone counterparts in any width. Some recent works [6, 73, 24] also explore anytime neural networks in multiple dimensions, e.g., depth, width, kernel size, etc.

Dynamic neural networks [63, 64, 46, 68] 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, 30, 64, 63, 44] achieve efficient inference in two ways, early exiting when shallower sub-networks have high classification confidence [4, 30, 44], or skipping residual blocks adaptively [64, 63]. Recently, **dynamic pruning methods** [29, 14, 9] using a variable subset of convolution filters have been studied. Channel Gating Neural Network [29] and FBS [14] identify and skip the unimportant input channels at runtime. In GaterNet [9], a separate gating network is used to predictively select the filters of the main network. Please refer to [19] for a more comprehensive review of dynamic neural networks.

**Weight sharing NAS** [5, 1, 3, 48, 7, 66, 18, 41, 6, 42], 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** [48, 7, 66], 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, 18, 41, 6, 42], 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 [63, 64, 46, 68, 4, 30, 44, 29, 14, 9], the dynamic routing agent and the main network are jointly trained, analogous to jointly optimized NAS methods [48, 7, 66]. Inspired by one-shot NAS methods [5, 1, 3, 18, 41], 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 slimmng gate and train the supernet with the IEB technique, then in Stage II, we fix the weights of the supernet and train the slimmng 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 [29, 14] and conditional convolution [68, 45], 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:

\[
\mathcal{Y} = \mathcal{W}_{\mathcal{A}(\theta, \mathcal{X})} \ast \mathcal{X},
\]

where \( \mathcal{W}_{\mathcal{A}(\theta, \mathcal{X})} \) represents the selected or generated input-dependent convolution filters. Here \( \ast \) is used to denote a matrix multiplication. Previous dynamic pruning methods [29, 14] 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
In recent years, a growing number of self-supervised methods with bootstrapping [16, 17, 8] and semi-supervised methods based on consistency regularization [37, 61] 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 $\theta$ and $\theta'$ denote the parameters of the online network and the target network, respectively. We have:

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

where $\alpha$ 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 [58, 59], 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 [72], 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 $Y$; $n$ sub-networks with random width are trained to predict the soft label generated by the widest sub-network of the target network, $Y'_L(\theta')$; the slimmest sub-network is trained to predict the probability ensemble of all the aforementioned sub-networks in the target network:

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

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

$$\begin{align*}
\mathcal{L}_E^{IEB}(\theta) &= \mathcal{L}_{CE}(Y_L(\theta), Y), \\
\mathcal{L}_R^{IEB}(\theta) &= \mathcal{L}_{CE}(Y_R(\theta), Y'_L(\theta)), \\
\mathcal{L}_S^{IEB}(\theta) &= \mathcal{L}_{CE}(Y_S(\theta), \overline{Y}'_{L,R}(\theta')).
\end{align*}$$

### 3.2. Dynamic Slimming Gate

In this section, we design the channel gate function $A(\theta, \mathcal{X})$ that generates the factor $\rho$ 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 $\rho$ 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

![Figure 4. Training process of slimmable supernet with In-place Ensemble Bootstrapping.](image-url)
choose the respective slimming ratio $\rho$ in a discrete candidate list vector $L_\rho$. 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 $X$ to a one-hot vector, we divide $\mathcal{A}(\theta, X)$ to two functions:

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

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 $X$ with dimension $\rho_{n-1}C_n \times H_n \times W_n$, $\mathcal{E}(X)$ reduces it to a vector $X_\mathcal{E} \in \mathbb{R}^{\rho_{n-1}C_n}$ which can be further mapped to a one-hot vector. By computing the dot product of this one-hot vector and $L_\rho$, we have the newly predicted slimming ratio:

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

Similar to prior works [28, 70] 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 $W_1 \in \mathbb{R}^{d \times C_n}$ and $W_2 \in \mathbb{R}^{g \times d}$ (where $d$ represents the hidden dimension and $g$ represents the number of candidate slimming ratio) and a ReLU non-linearity layer $\sigma$ 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:

$$\mathcal{F}(X_\mathcal{E}) = \text{argmax}(W_2 \sigma(W_1[:, :\rho_{n-1}C_n](X_\mathcal{E}))). \quad (8)$$

Note that input $X$ with dynamic channel number $\rho \times C$ is projected to a vector with fixed length by the dynamically sliced weight $W_1[:, :\rho_{n-1}C_n]$. Our proposed channel gating function has a similar form with recent channel attention methods [28, 70]. The attention mechanism can be integrated into our gate with nearly zero cost, by adding another fully connected layer with weights $W_3$ 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:

$$\tilde{X} = X \odot \delta(W_3[:, :\rho_{n-1}C_n](\sigma(W_1[:, :\rho_{n-1}C_n](X)))) . \quad (9)$$

where $\delta(x) = 1 + \tanh(x)$ 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 $\mathcal{L}_{dp\times}$ 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 $L_{cls}$, we use gumbel-softmax [34], 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 [34] 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 [72], 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 $X_{\text{easy}}$ that can be correctly classified by the slimmest sub-network; b) Hard samples $X_{\text{hard}}$ that can not be correctly classified by the widest sub-network; c) Dependent samples $X_{\text{dep}}$. 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 $T(X_{\text{easy}}) = [1, 0, \ldots, 0]$). 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. $T(X_{\text{hard}}) = T(X_{\text{dep}}) = [0, \ldots, 0, 1]$). 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:

$$\mathcal{L}_{SGS} = T_{\text{slim}}(X) \ast \mathcal{L}_{CE}(X, T(X_{\text{easy}})) + (-T_{\text{slim}}(X)) \ast \mathcal{L}_{CE}(X, T(X_{\text{hard}})) \quad (10)$$

where $T_{\text{slim}}(X) \in \{0, 1\}$ represents whether $X$ is truly predicted by the slimmest sub-network and $\mathcal{L}_{CE}(X, T) = -\sum T \ast \log(X)$ 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 [12] and CIFAR-10 [36]) and a standard object detection dataset (i.e., PASCAL VOC [13]). 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 \times 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 [20] and the lightweight non-residual network MobileNetV1 [26], to evaluate our method.

In Dynamic Slimmable ResNet 50 (DS-ResNet), we insert our double-headed gate in the beginning 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. $1 \times 1$ 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 [74, 72].

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 [74, 72], guaranteeing itself the practicality to use batch normalization. Following [72], 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 [74, 72, 71]. 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 $0.9 \times$ 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 [35] and [32]. 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 [49] and [47] 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 $2.4 \times$ computation reduction over ResNet-50 (76.1% [20]) and MobileNetV1 (70.9% [26]) 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 $1.31 \times$ and $1.91 \times$, our DS-MBNet achieves comparable $1.17 \times$ and $1.62 \times$ acceleration with much higher performance. In particular, our DS-MBNet surpasses the original and the channel scaled MobileNetV1 [26] 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 the SOTA pruning methods EagleEye [40] and Meta-Pruning [51] 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 [29] by 2.5%. Fourth, our DS-Net also consistently outperforms its static counterparts. Our DS-MBNet-S surpasses AutoSlim [71] and US-Net [72] 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 learning 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 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) [47]. 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 dimension by the feature transform module in FSSD [47].

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 analyze the effect of IEB technique with MobileNetV1. We train a Slimmable MobileNetV1 supernet with three settings: original in-place distillation, in-place distillation with 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 stabilizing 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 $\mathcal{L}_{cpx}$, 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 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. $\mathcal{T}(X_{hard}) = [0, \ldots, 0, 1]$). (ii) Give Up: Push the hard samples to use the slimmest path to save computation cost (i.e. $\mathcal{T}(X_{hard}) = [1, 0, \ldots, 0]$). In both of the strategies, dependent samples are encouraged to use the widest sub-network (i.e. $\mathcal{T}(X_{dependent}) = [0, \ldots, 0, 1]$). 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 attributed 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-M. The height of those colored blocks illustrate the path of input samples that are routed to the sub-networks with respective slimming ratio $\rho$.

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× computation reduction and 1.62× 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. RCYX2020071414462083), Shenzhen Basic Research Project (Project No. JCYJ20190807142211365), Leading Innovation Team of the Zhejiang Province (2018R01017) Project (Project No. JCYJ20190807154211365), Leading Shenzhen Outstanding Youth Research Project (Project No. RCYX20200714114642083), Shenzhen Basic Research Project (Project No. RCYX2020071414462083), and can generalize well on CIFAR-10 classification task and VOC object detection task.

References
[1] Youhe Akimoto, Shinichi Shirakawa, Nozoumu 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. arXiv:1908.09791, 2019. 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] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In CVPR, 2019. 12
[12] 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
[13] 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
[14] 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
[15] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Łukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv:1706.02677, 2017. 12
[16] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad GheSLaghli Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. arXiv:2006.07733, 2020. 4
[17] Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-bastien Grill, Florent Altché, Rémi Munos, and Mohammad GhesLaghli Azar. Bootstrap latent-predictive representations for multilabel reinforcement learning. In ICML, 2020. 4
[18] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. arXiv:1904.00420, 2019. 1, 3
[19] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. arXiv:2102.04906, 2021. 3
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016. 3, 6, 7
[21] Yang He, Guoiliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In IJCAI, 2018. 1, 6
[22] Yihui He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Channel pruning for accelerating very deep neural networks. In ICCV, 2017. 1, 6
[23] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distillation. arXiv:1503.02531, 2015. 1, 3
[24] Daniel G. Hong, Hoang Chau, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V Le, and Hartwig Adam. Searching for mobilenetv3. arXiv:1905.02244, 2019. 12
[25] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Welstead, Marco An-
detection, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861, 2017. 1, 6, 7
[27] Hanzhang Hu, Debadeepta Dey, Martial Hebert, and J Andrew Bagnell. Learning anytime predictions in neural networks via adaptive loss balancing. In AAAI, 2019. 3
[28] J Hu, L Shen, S Albanie, G Sun, and E Wu. Squeeze-and-excitation networks. IEEE transactions on pattern analysis and machine intelligence, 2019. 5, 12
[29] 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, 7
[30] 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, 3, 6
[31] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In CVPR, 2017. 3
[32] 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
[33] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Improving convolutional neural networks for efficient integer-arithmetic-only inference. In CVPR, 2018. 1
[34] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In NeurIPS, 2017. 5
[35] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In CVPR, 2018. 6, 7
[36] A. Krizhevsky and G. Hinton. Learning multiple layers of feature representations from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009. 5
[37] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. arXiv:1610.02242, 2016. 4
[38] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In ICLR, 2017. 3, 12
[39] Hankook Lee and Jinwoo Shin. Anytime neural prediction via slicing networks vertically. arXiv:1807.02609, 2018. 3
[40] Bailin Li, Bowen Wu, Jiang Su, Guanrun Wang, and Liang Lin. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In ECCV, 2020. 6
[41] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guanrun Wang, Xiaodan Liang, Liang Lin, and Xiaojuan Chang. Block-wisely supervised neural architecture search with knowledge distillation. In CVPR, 2020. 1, 3
[42] Changlin Li, Tao Tang, Guanrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojuan Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. arXiv:2103.12424, 2021. 3
[43] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. arXiv:1608.08710, 2016. 1
[44] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In ICCV, 2019. 1, 3
[45] 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
[46] 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
[47] Zuoqin Li and Fuqiang Zhou. Fssd: feature fusion single shot multibox detector. arXiv:1712.00960, 2017. 6, 7
[48] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In ICLR, 2019. 3
[49] 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
[50] Zhang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In ICCV, 2017. 1
[51] 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
[52] Zhang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In ICLR, 2019. 6
[53] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In ICCV, 2017. 1, 6
[54] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. In ICLR, 2017. 12
[55] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaoqian Jiang, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. ACM Computing Surveys, 2021. 1
[56] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In ICLR, 2015. 1
[57] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In CVPR, 2018. 1, 12
[58] Zhiqiang Shen and Marios Savvides. Meal v2: Boosting dynamic convolution via matrix decomposition. In NeurIPS, 2021. 3
[59] 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
[62] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSEMA: Neural networks for machine learning, 4(2):26–31, 2012.

[63] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In ECCV, 2018.

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

[65] 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.

[66] 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.

[67] Yuxin Wu and Kaiming He. Group normalization. In ECCV, 2018.

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

[69] 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.

[70] Zongxin Yang, Linchao Zhu, Yu Wu, and Yi Yang. Gated channel transformation for visual recognition. arXiv:1909.11519, 2019.

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

[72] Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In ICCV, 2019.

[73] 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.

[74] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas S. Huang. Slimmable neural networks. In ICLR, 2019.

[75] 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.

[76] 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.

[77] 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.
Appendix

A. Implementation Details

**Losses in Stage II.** Complexity penalty loss $\mathcal{L}_{cplx}$ is used to increase the model efficiency in training stage II. To provide a stable and fair constraint, we use the number of multiply-adds on the fly, $\text{MAdds}(X, \theta)$, as the metrics of model complexity. Specifically, the complexity penalty is given by:

$$\mathcal{L}_{cplx}(X, \theta) = \left( \frac{\text{MAdds}(X, \theta)}{T} \right)^2,$$

where $T$ is a normalize factor set to the total MAdds of the supernet in our implementation. Note that this loss term always pushes the gate to route towards a faster architecture, towards an architecture with target MAdds, which can effectively prevent routing easy and hard instances to the same architecture.

Overall, the slimming gate can be optimized with a joint loss function:

$$\mathcal{L}(X, \theta) = \lambda_1 \mathcal{L}_{cls} + \lambda_2 \mathcal{L}_{cplx} + \lambda_3 \mathcal{L}_{SGS}.$$  \hfill (12)

The three balancing factors are set to $\lambda_1 = 1, \lambda_2 = 0.5, \lambda_3 = 1$ in our experiments. Different target MAdds is reached by adjusting the routing space during gate training. Note for instance, when training the gate of DS-MBNet-S, we set $\rho \in [0.35 : 0.05 : 0.5]$ to prevent routing to heavier sub-networks.

**Equispaced channel group.** Following previous works [72, 71], we set the the smallest division of channel number to 8. When using 0.05 as the interval of $\rho$, rounding channels by 8 may result in different intervals, which could lead to training failure when using Group Normalization [67]. To prevent such problem, we always adopt a consistent interval (e.g. 8, 16, 32) in a single layer, instead of multiplying $\rho$ and rounding the channel. This results in a difference of the slimming ratio between our implemented architecture and our design.

**Additional details.** Weight decay is set to $1^{-4}$ in all of our experiments on ImageNet. To stabilize the optimization, weight decay of all the layers in the dynamic gate is removed. The weight $\gamma$ of the last normalization layer of each residual block is initialized to zeros following [73]. The weight of the fully-connected layer in channel attention head, $W_3$ in Eqn. 8, is also zero-initialized to ease the following [70]. Additional training techniques include [15, 11]. We do not use label smoothing [54], DropPath [38] and RMSProp [62], which are popularly used in previous works [60, 25, 71, 72].

B. Experiments on EfficientNet

We also applied our method on EfficientNet [60], a state-of-the-art network family with high efficiency. Similar to our DS-MBNet, Dynamic Slimmable EfficientNet-B0 (DS-EffNet) has only one slimming gate after its 8-th inverted residual block, controlling the rest 8 blocks. The fixed slimming ratio for the first 8 blocks is 0.5, while a uniform dynamic slimming ratio $\rho \in [0.75 : 0.05 : 1.75]$ is used for the last 8 blocks. This supernet with 20 paths in total is trained with a similar config with the supernet of DS-ResNet and DS-MBNet.

We train the supernet with 512 total batch size using 0.2 learning rate that decays with a cosine scheduler in 150 epochs. To enable direct comparision, we opt to reproduce the EfficientNet results using our training setup, with a 150 epoch schedule and no extra enhancement of DropPath [38], RMSProp [62], etc.

The result is shown in Tab. 8. DS-EffNet outperforms the original EfficientNet-B0 by 0.7% and 0.8%, proving its efficacy on recent methods with inverted bottleneck blocks [57] and Squeeze-and-Excitation module [28].

C. Additional Ablations

**Slimming gate.** We analysis the improvement brought by slimming gate by comparing the performance of DS-Net and its supernet. As shown in Tab. 9, slimming gate boosts the performance of DS-MBNet-S and DS-ResNet-S by 0.8% and 1.2% respectively, comparing to sub-networks with similar sizes in their supernet.

**Distillation temperature.** Temperature $\tau$ in distillation loss was first introduced in [23] to control the smoothness of the target. Using a properly larger $\tau$ usually yields better performance of the student. Surprisingly, we find a huge performance degradation in the slimmest sub-network when using larger $\tau$ in in-place distillation. We test $\tau = 4$ with DS-MBNet for 40 epochs and compare the it with the performance of default setting ($\tau = 1$). As shown in Tab. 10, the performance of the slimmest sub-network decrease by 10.2% after applying the temperature $\tau = 4$. 

---

| Method          | MAdds | Top-1 Acc. |
|-----------------|-------|------------|
| DS-EffNet-B0    | 399M  | 76.0       |
| DS-EffNet-L     | 400M  | 76.7       |
| DS-EffNet-S (Ours) | 267M  | 74.6       |
| DS-ResNet-S (Ours) | 270M  | 75.4       |

Table 9. Ablation analysis of slimming gate.

| model                      | MAdds | Top-1 Acc. |
|---------------------------|-------|------------|
| supernet (DS-MBNet)       | 140M  | 69.3       |
| DS-MBNet-S                | 153M  | 70.1       |
| supernet (DS-ResNet)      | 1.1B  | 73.4       |
| DS-ResNet-S               | 1.2B  | 74.6       |

Table 10. Ablation analysis of distillation temperature $\tau$ (40 epochs).